University of Southern Queensland
Faculty of Engineering and Surveying

Development of a Tournament Management System

A dissertation submitted by

Benjamin Brett Thomas

In fulfillment of the requirements of

Courses ENG4111 and ENG4112 Research Project

Towards the degrees of

Bachelor of Engineering (Computer Systems)
Bachelor of I.T. (Applied Computer Science)

Submitted: January, 2010

ABSTRACT

Online gaming competitions are growing exponentially in popularity as internet infrastructure improves. Effectively managing the simplest tournament and tournament assets has become a very time consuming task. This dissertation examines methods for automating some of the most common tournament management tasks including; scheduling, identity authentication, bracket building and server management.

This document develops separate models for completing these common tasks and examines tools already available for tournament management. A number of programming languages and environments are used throughout in order to effectively create the simplest module model. Background research in tournament formats, communication protocols and game software form the basis of module design and development.

This dissertation develops concepts and ideas that form the basis of a programming solution to tournament management. These ideas are further developed thought the implementation and testing of automated modules and an insight into future study available in this field.

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 Research Project Part 1 &
ENG4112 Research Project Part 2

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course "Project and Dissertation" is to contribute to the overall education within the student's chosen degree programme. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Professor Frank Bullen
Dean
Faculty of Engineering and Surveying

CANDIDATES CERTIFICATION

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Benjamin Brett Thomas
Student Number: Q10225151

_____
Signature

_____
Date

ACKNOWLEDGEMENTS

This research project was carried out under the principle supervision of Dr Alexander Kist

Appreciation is also due to:

Brendan Anabel

Alex Mottshaw

Tom Fabian

Joel White

Rob Skillington

Contents

ABSTRACT I

LIMITATIONS OF USE II

CANDIDATES CERTIFICATION III

ACKNOWLEDGEMENTS IV

LIST OF FIGURES VIII

LIST OF TABLES IX

ABBREVIATIONS X

CHAPTER 1 1

1.1 Outline of the study 1

1.2 Introduction 2

1.3 The Problem 3

1.4 Scope 4

1.5 Research Objectives 5

1.6 Conclusions 6

CHAPTER 2 7

2.1 Introduction 7

2.2 E-Sports 8
2.2.1 Professional E-Sports 8
2.2.2 GotGames 9
2.2.3 Online Environment 9

2.3 Game and Platform Software 11
2.3.1 Valve 11
2.3.2 Steam Platform 11
2.3.3 Source Engine 12
2.3.4 Dedicated Servers and TV Relays 13
2.3.5 RCON 14

2.4 Tournament Formats **15**
2.4.1 Seeding and Draws 15
2.4.2 Single Elimination 15
2.4.3 Double Elimination 16
2.4.4 League 18
2.4.5 Group 18

2.5 Existing Tournament Management Software **19**
2.5.1 The CEVO System 19
2.5.2 Bracket Maker 23

2.5 Information Feeds and Protocols **25**
2.5.1 JSON 25
2.5.2 XML 26
2.5.3 Direct Database Access 28

2.6 Summary **29**

CHAPTER 3 **30**

3.1 Introduction **30**

3.2 System Overview **31**
3.2.1 GotGames Resources and Infrastructure 31
3.2.2 Language and Protocol Choices 33

3.3 RCON Interface Module **35**
3.3.1 RCON Protocol 35
3.3.2 Sending Packets 35
3.3.3 Receiving Packets 36
3.3.4 RCON Design and Development 37

3.4 Client Software **40**
3.4.1 Client Functions 41
3.4.2 Automated Scheduling and Bracket Functions 45

3.5 Summary **50**

CHAPTER 4 **51**

4.1 Introduction **51**

4.2 Test Procedures **52**
4.2.1 RCON Interface 52
4.2.2 Client and Chat Server 53
4.2.3 Auto-Scheduler 56
4.2.4 Bracket Builder 57

4.3 Summary **58**

CHAPTER 5 **59**

5.1 Introduction **59**

5.2 Security **60**

5.3 Conclusions **62**

5.4 Further Work **63**

REFERENCES **64**

APPENDICES **66**

LIST OF FIGURES

Figure 1 – Manual tournament management. 2
Figure 2 - Simplified system model 4
Figure 3 - Single elimination bracket 16
Figure 4 - Double elimination bracket 17
Figure 5 – CEVO server management 20
Figure 6 – CEVO match listing 21
Figure 7 – CEVO client 22
Figure 8 – Bracket Maker example 24
Figure 9 - Broadcast module 25
Figure 10 – RCON interface 38
Figure 11 – Client/Server module 40
Figure 12 - Scheduling and bracket building 45
Figure 13 – Bracket layout 48
Figure 14 – Client login screen 53
Figure 15 – The first client 54
Figure 16 – The second client 54
Figure 17 - Createserver.php display 56
Figure 18 - Bracket Builder output 57

LIST OF TABLES

Table 1 - JSON Characteristics 26
Table 2 - XML Characteristics 27
Table 3 - Language choices 34
Table 4 - SERVERDATA_AUTH_RESPONSE codes 36
Table 5 - SERVERDATA_RESPONSE_VALUE codes 37

ABBREVIATIONS

API	Application Programming Interface
CEVO	Cyber Evolution
E-sports	Electronic Sports
FTP	File Transfer Protocol
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
PC	Personal Computer
PHP	Hypertext Preprocessor
RCON	Remote Control Protocol
SQL	Structured Query Language
Steam ID	Steam Identification Code
STV	Source TV
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Outline of the study

The need for GotGames to derive an automated tournament management system has become apparent over recent years as tournament sizes have increased at alarming rates. Current manual techniques are very time consuming and divert staff resources from other high priority tasks. The purpose and scope of this study is detailed in section 1.4 Research Objectives.

1.2 Introduction

GotGames in an Australian based e-sports company focused on delivering high quality gaming competitions in both an online and offline environment. Although the services offered by GotGames cover a large number of platforms and game environments, the largest, most lucrative and popular competitions are PC based and run on Valve's Source Engine.

Current methods for managing game servers, match scheduling and result reporting all rely on staff manually manipulating the resources and data, and results in excessive hours wasted on often tedious and repetitive tasks. Figure 1 illustrates the current situation where all aspects of tournament management are manual.

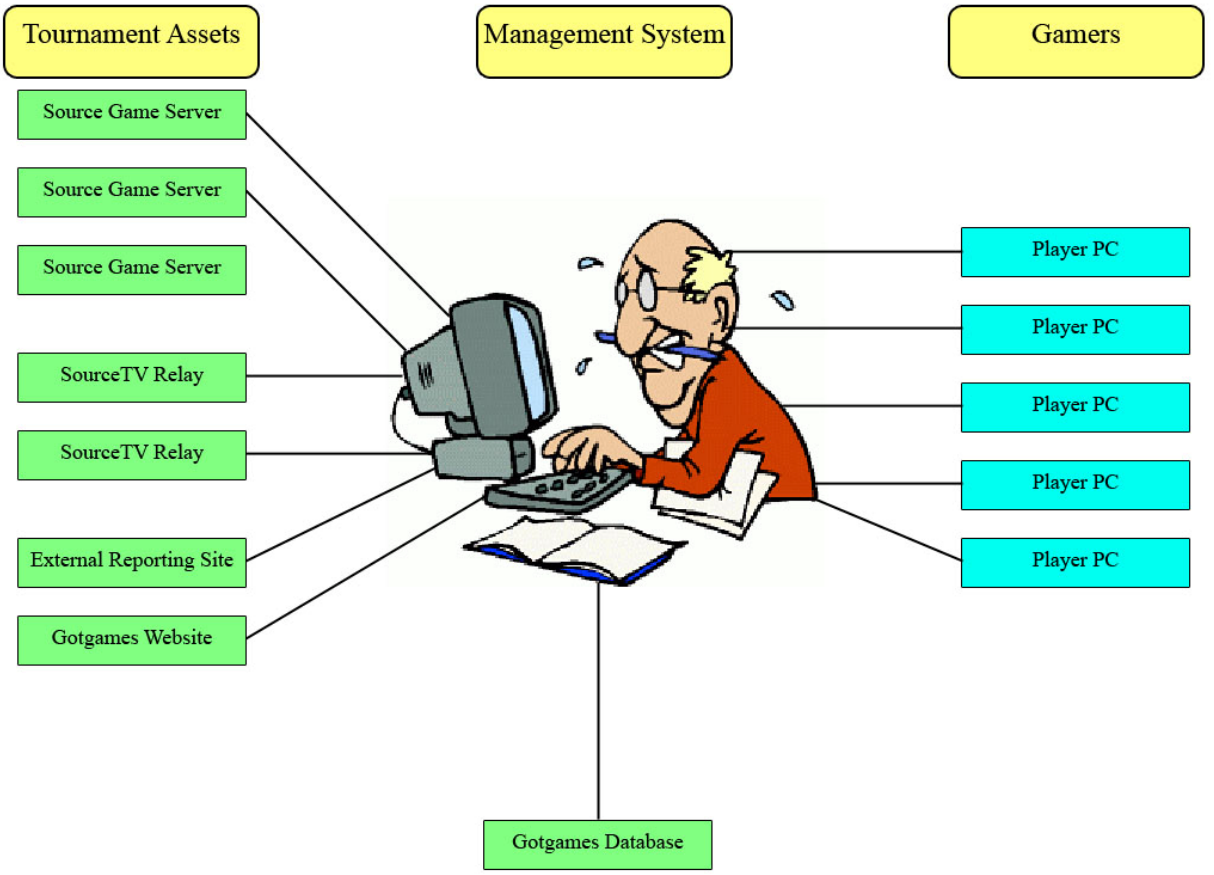

Figure 1 – Manual tournament management.

1.3 The Problem

Despite the obvious need to automate a complete system as discussed in section 1.2 Introduction the timeframe allocated for this project required that the scope be reduced to designing, researching and developing important modules that will eventually become part of an overall solution. The specific areas of importance that will be addressed through this dissertation are:

- Broadcast methods and formats for sharing competition information.
- Controlling remote game and spectator servers and automation of this control.
- Determining the identity of a player given the ambiguous nature of the online environment.
- Automating scheduling and the associated tournament resource management.
- Displaying spectator and scheduling information in a bracket format.

Another aspect important to the development of this system is the problem that stems from the ambiguous nature of online environments. Online e-sports competitions often have sizeable sums of cash prizes and it is important that the identity of each competitor can be confirmed during all aspects of the competition.

1.4 Scope

As mentioned in section 1.3 The Problem, the scope of this work has been reduced to include the design and development of important system modules and not an entire system. Research has also been limited to subjects that directly impact the design and development phases. Although outside the scope of this work developing a comprehensive system would require the outcome of this dissertation as integral modules.

The principles and techniques used in this project are intended for use beyond the limitations of the PC and the Source Engine however this is work for future academics which is addressed in Section 5.4 Future Work. Figure 2 demonstrates a simplified view of the developed tournament management system.

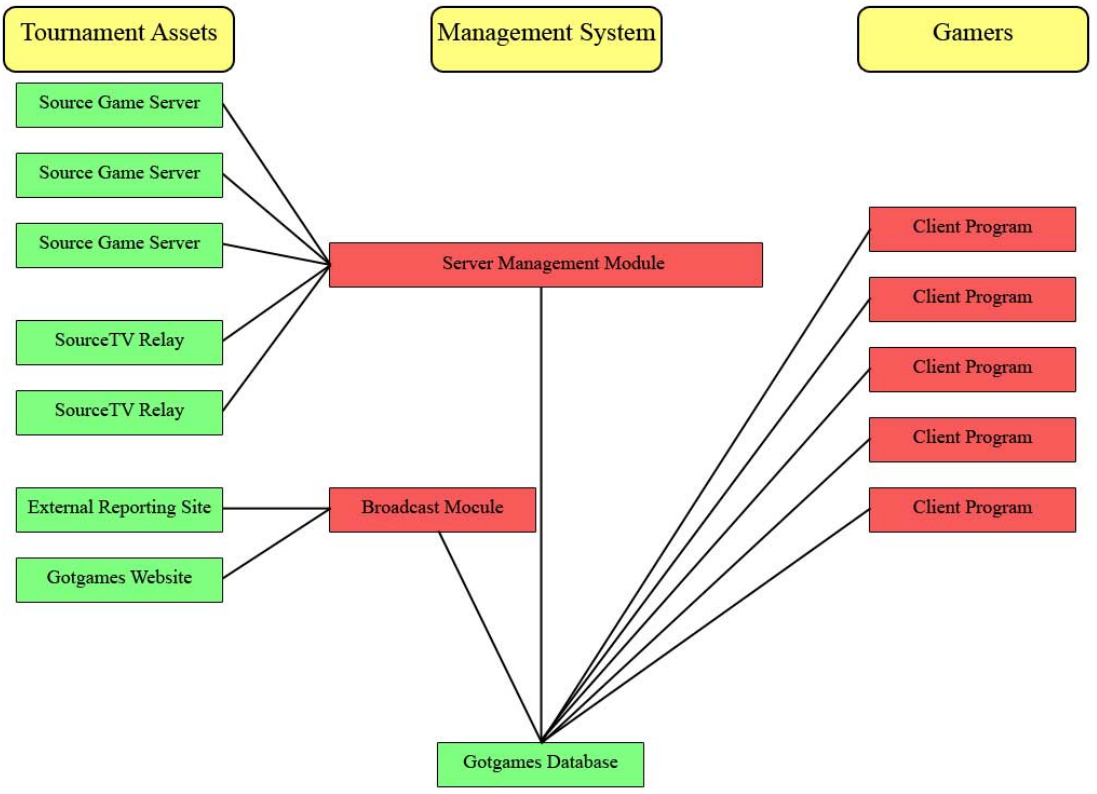

Figure 2 - Simplified system model

1.5 Research Objectives

This research comprised of identifying problems, evaluating existing solutions and investigating protocols and interfaces within the field of online tournament management specifically related to the Source Engine. The aim of this research was to provide:

- Solid understanding and limitation of existing solutions.
- Determine appropriate formats to broadcast tournament data.
- Provide background information and definitions for e-sports, the Steam platform and the Source Engine.
- Investigate the RCON protocol to facilitate game server control.
- Investigate Steam install environments in order to effectively design a client program.

The research will provide solid background knowledge of all the elements involved in this project as well as focus on those areas that are important in the design and development of tournament management modules.

1.6 Conclusions

This dissertation aims to identify key aspects of online tournament management and provide a solid foundation for the design and development of tournament management modules that can be integrated into existing and future systems. The research is expected to result in evidence that automation will not only save valuable time but also that existing systems are not adequate for GotGames use. A review of literature for this research will identify communication protocols, features and security mechanisms that can be used in the module design and development. The outcomes of this study will be used for the design and development of server management modules, client identification software and server based software that will be used in existing GotGames infrastructure.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will review literature to establish the need for automation of tournament systems as well as provide background knowledge essential to the design and development phase discussed in section 3 Module Design and Development. Once the foundations are covered this chapter will consider current tournament management options and software and take a closer look at some of the functions and protocols that will be used throughout this work.

2.2 E-Sports

E-sports or ‘electronic-sports’ is a term covering all forms of competitive gaming across all platforms encompassing both offline and online environments. This section will establish the local e-sports community and infrastructure by examining project sponsor GotGames after establishing a quick overview of the local e-sports scene.

2.2.1 Professional E-Sports

No matter the activity if it is possible to make it competitive then there is little doubt that people will. E-sports has long been portrayed as a basement activity for the rare stereotypical geek community but recent events and innovations have seen e-sports on major television stations throughout USA, South America, Asia and Europe. Being thrust into the spotlight and offering prize pools in excess of millions of dollars (CGS 2008) has seen the popularity of e-sports soar.

One of the most appealing attributes of e-sports is its low entry cost and the ability for everyday users to compete over the internet in the comfort of their own homes. Online tournaments and competitions have quickly become the norm and have seen massive growth over the past 3 years; yearly growth figures in excess of 300% are not uncommon for many games within Australia (Mottshaw, unpub).

Along with solid positive growth the e-sports scene in Australia and internationally has seen a huge resurgence in online competitions with improved internet speed and games equipped with lag compensation. Moving the competitive environment to the player’s home adds an element of ambiguity and emphasis the need for identity authentication throughout online competitions.

2.2.2 GotGames

GotGames (www.gotgames.com.au) is an Australian company specializing in online e-sport tournament management. Although in recent years GotGames have branched into many platforms and game titles the main focus of the company is Source based games and providing for their huge local following.

In order to effectively deal with large increases in player numbers GotGames has sponsored this work and will implement parts of this project as discussed in Section 3 Module Design and Development. GotGames has a number of assets available for the development and testing phases of this work including:

- Access to existing user data stored in SQL databases.
- Dedicated game servers deployed within both Linux and Windows environments.
- Dedicated spectator relays deployed within the Linux environment.
- Access to existing tournament data for testing.
- Access to web server deployed on a Linux machine.

2.2.3 Online Environment

The successful outcome of this work will result in modules designed to be implemented in an online tournament management system. Although in theory many features and options within these modules can also be utilized for offline competitions the design and development will concentrate purely on an online environment. Online tournaments are managed remotely and can cater for large numbers of teams and players; therefore automation of these systems offers the greatest gains from a corporate perspective.

In an online environment player authentication becomes a major issue facing tournament organizers and administrators. Each and every player must be who they claim to be and not representing a false identity or alternative identity throughout the competition. There

are many features of the game and platform software that can assist tournament organizers and administrators in identifying players and these are discussed further in the next section.

2.3 Game and Platform Software

This section will examine the software used by gamers and organizers within GotGames focusing on the Source Engine and associated servers and deployment platforms. This is the software that the management modules will manipulate in order to achieve the goals of:

- Manipulate dedicated game servers.
- Authenticate player's identities.
- Manipulate dedicated spectator relays.
- Gather user information through client software.

2.3.1 Valve

"Valve is an entertainment software and technology company founded in 1996 and based in Bellevue, Washington. The company's debut title, Half-Life®, has won over 50 Game of the Year Awards and was named "Best PC Game Ever" in the November 1999, October 2001, and April 2005 issues of PC Gamer, the world's best-selling PC games magazine." (valve, 2008)

Valve is responsible for all the software covered in this section and is a developer of leading-edge technologies including the Source game engine and Steam, a leading platform for digital content with over 10 million registered users.

2.3.2 Steam Platform

Steam is the name of Valve's digital distribution platform and has a worldwide user base in excess of 20 million (valve 2010) (internet café accounts included). In order to play licensed games such as those using the Source Engine each player must have Steam installed and operating on their PC.

Steam is an account based virtual file system that allows users to access programs including games only once they have authenticated with a username and password login

system. In this way users can have access to programs that have been purchased through Steam on any PC around the world given that once they login the required data files are downloaded.

The most useful feature of the Steam platform from this project's perspective is that every account is assigned a unique Steam ID. This Steam ID is used to identify the account and its owner and can be access from within games or directly from the steam logs. By utilizing this unique Steam ID one can successfully link players to various accounts thus forming the basis of a user authentication process. Features of the Steam platform include:

- Integrated friend and user chat system.
- Ability for users to modify games and programs with limited source code available.
- Steam Cloud stores player's settings so that no matter where they login from their game settings are always consistent.
- Accounts are linked to unique Steam IDs.
- Integrated store allowing online shopping and content download.
- Automated updates.
- Online authentication required to run the steam process.

It is important to emphasize that it is necessary to validate every Steam game online before it can be played, although an offline mode is available. There are no alternate methods of activation such as via telephone or fax, which causes the system to deny access to those without Internet connections.

2.3.3 Source Engine

The Source Engine is a 3D game engine developed by valve and is the workhorse behind a large number of competitive games including Counter-Strike Source and Team Fortress 2. The Engine has been constantly upgraded since its release in June 2004 and licensed to

a number of platforms including Microsoft Windows, Xbox, Xbox 360 and Playstation 3. The Engine itself is not a particularly important part of tournament management and is concerned with how the game client's interact with the software interface and the players overall visual experience. With the focus of this work being on tournament management it is important that the game clients be touched on briefly, however the following section 2.3.4 Dedicated Servers and Spectator Relays will introduce important resources for management systems.

2.3.4 Dedicated Servers and TV Relays

As with the large majority of online games, those based on the Source Engine operate in a client server model. This means that for every match there must be a server that is capable of accepting the client connections and controlling the game interface, physics and interaction. For online tournament play these servers are dedicated (server only software) and can be deployed in either a Microsoft Windows or Linux environment. Source dedicated servers are command line based and it is not uncommon to see a number of different servers located on a single machine.

Along with dedicated servers Valve have developed a dedicated spectator service called Source TV that enables clients to view a match from within the game client without being able to influence or affect game play. In order to reduce the burden on resources these Source TV services operate in much the same way as a game server and can operate from multiple locations while being directed to a single dedicated Source server. The delocalization of Source TV leads to the term relay being used and in effect Source TV binaries can be infinitely chained to allow for any spectator numbers.

The tournament management models will need to interface with these Server and Source TV binaries in order to manipulate the following:

- Restart servers for scheduled matches.
- Set server connection passwords for clients.

- Change server maps.
- Point Source TV relays to the appropriate server.
- Set predetermined rule sets through server variables.

Valve has implemented a remote administration service that can be used to manipulate Source Servers and Source TV Services. This remote administration service is called RCON and is covered in the next section.

2.3.5 RCON

RCON is a remote administration service that can be used to control all aspects of Source Servers and Source TVs including setup, game environment, client connection parameters and blocking client commands. In order to access RCON commands users must first authenticate themselves using the RCON password. The RCON password is set during the installation and setup of the Source Server or Source TV Server and can only be changed by those with FTP access to the server machine.

Once a user authenticates themselves with the RCON password commands can be wrapped into packets and sent to the required server. The developed tournament management modules can effectively control any Server or Relay through the use of the RCON interface module that is discussed in-depth in Section 3.3.

2.4 Tournament Formats

Section 2.4 will focus on available tournament formats and examine how brackets are constructed and seeding is calculated.

2.4.1 Seeding and Draws

All e-sports competition brackets and draws are populated either by seeding or random drawing. As with all sports seeding is the preferred method giving teams with higher seeds early match ups against lower seeded teams. However in such a volatile and ever changing environment it is not always realistic to seed teams and therefore any automated system will also have to accommodate the option to randomly seed teams.

Scheduling modules will need to take into account options for random seeding and set seeding in order to fulfill the requirements of GotGames.

2.4.2 Single Elimination

Single elimination is commonly used for tournaments where an aggressive timeline is an important factor. Teams or players are eliminated after a single loss and the bracket only guarantees that the winning team is the best. However without seeding single elimination brackets cannot guarantee that the best teams make it to the final as one of the finalist may have knocked out a better team earlier in the bracket.

It is important to note that the number of teams in a single elimination bracket must be a multiple of 2 to avoid byes being recorded. Due to limited resources GotGames competitions of this type have always been full and therefore a simplistic approach of avoiding byes can be used in the auto-scheduling module discussed in Section 3.4.2.

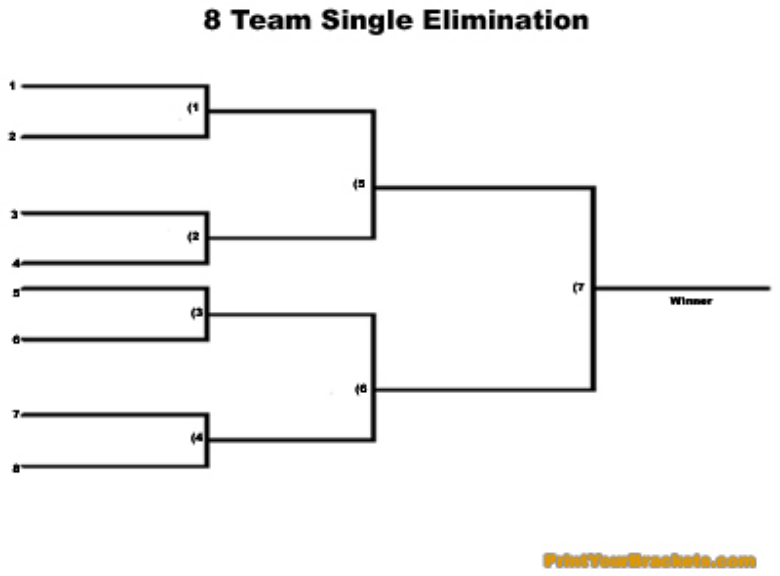

Figure 3 - Single elimination bracket

The number of matches required for a single elimination bracket can be calculated using the formula:

$$N - 1 = Total\ Matches$$
$$where\ N\ is\ the\ total\ number\ of\ teams.$$

2.4.3 Double Elimination

Double elimination is the preferred format for tournaments of short length, the bracket guarantees that the top teams progress to the later matches. The tournament bracket consists of a winners and a losers section with teams dropping to the lower bracket upon a single loss and two losses eliminating them from the tournament.

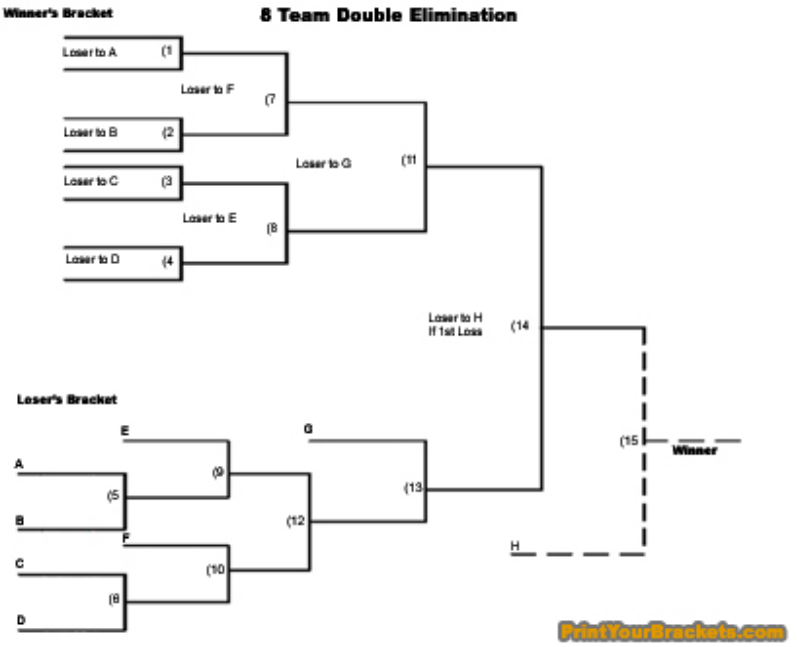

Figure 4 - Double elimination bracket

As with the single elimination bracket double elimination brackets require that the total number of teams be a multiple of 2 in order to avoid byes. The main advantage of a double elimination bracket over that of a single elimination is that it guarantees that the best two teams will be competing in the final.

The number of matches required for a single elimination bracket can be calculated using the formula:

$$\big((N-1)\,x\,2\big)\,+1 = Total\ Matches$$
$$where\ N\ is\ the\ total\ number\ of\ teams.$$

As demonstrated by the above equation a double elimination bracket requires more than twice as many matches as the equivalent single elimination bracket.

2.4.4 League

League formats pit all the teams into a single group and matches are played in a round robin format. This format is used for tournaments that take place over an extended period of time and are restricted to a relatively small numbers of teams. Points are awarded for win, draw and loss results and like group stage tournaments it is common for the top teams at the end of the league season to progress to a double or single elimination bracket.

Because the league format is only suitable for a limited number of teams competing over an extended timeframe it is not the source of a lot of invested employee time at GotGames. For this reason and the fact that the initial setup and scheduling is done for a number of weeks as opposed for a single day, league play will be omitted from this project and will be a feature that future work could implement.

2.4.5 Group

A favorite of most big tournaments, group stages splits teams up into a number of smaller pools were round robin games are played. During group stages points are awarded for a win, draw or loss and the resulting table is used to assess the teams that will advance from each group. Seeding is used to ensure that the better teams are evenly distributed between the groups and the top teams from each group are then placed into a single or double elimination finals bracket.

Like league brackets group stages are relatively easy to manually manage and are seldom used in GotGames competitions. For this reason group stage formats will be omitted from the management modules.

2.5 Existing Tournament Management Software

This section will examine existing software options for tournament management open to GotGames. This section will examine the features of each and their suitability for implementation by GotGames as an overall solution. In particular this section will focus on the CEVO proprietary system and free options available such as Bracket Maker.

2.5.1 The CEVO System

Comprehensive packages for online tournament management are not widely available however there are a number of companies that operate their own custom systems. One of the biggest and most successful of which is the North American based company, CEVO (Cyber Evolution). Although differing in that the CEVO system is designed specifically to run internal competitions it has been adapted to allow partnered companies to utilize some of the tournament management functions.

The CEVO system consists of 2 parts; a frontend that allows competitors to establish themselves in teams and enter personal information including Steam IDs; and a backend that allows administrators to control player and tournament information. A similar frontend already exists with the current GotGames system and therefore will not be addressed within this work. This project focuses on the backend and the automation of tasks associated with server management, match scheduling and client authentication.

Like the proposed system the CEVO system allows administrators to control a number of options within the server management section. However the CEVO system requires that servers be manually assigned to each match and the system is limited to servers that are owned and operated by CEVO.

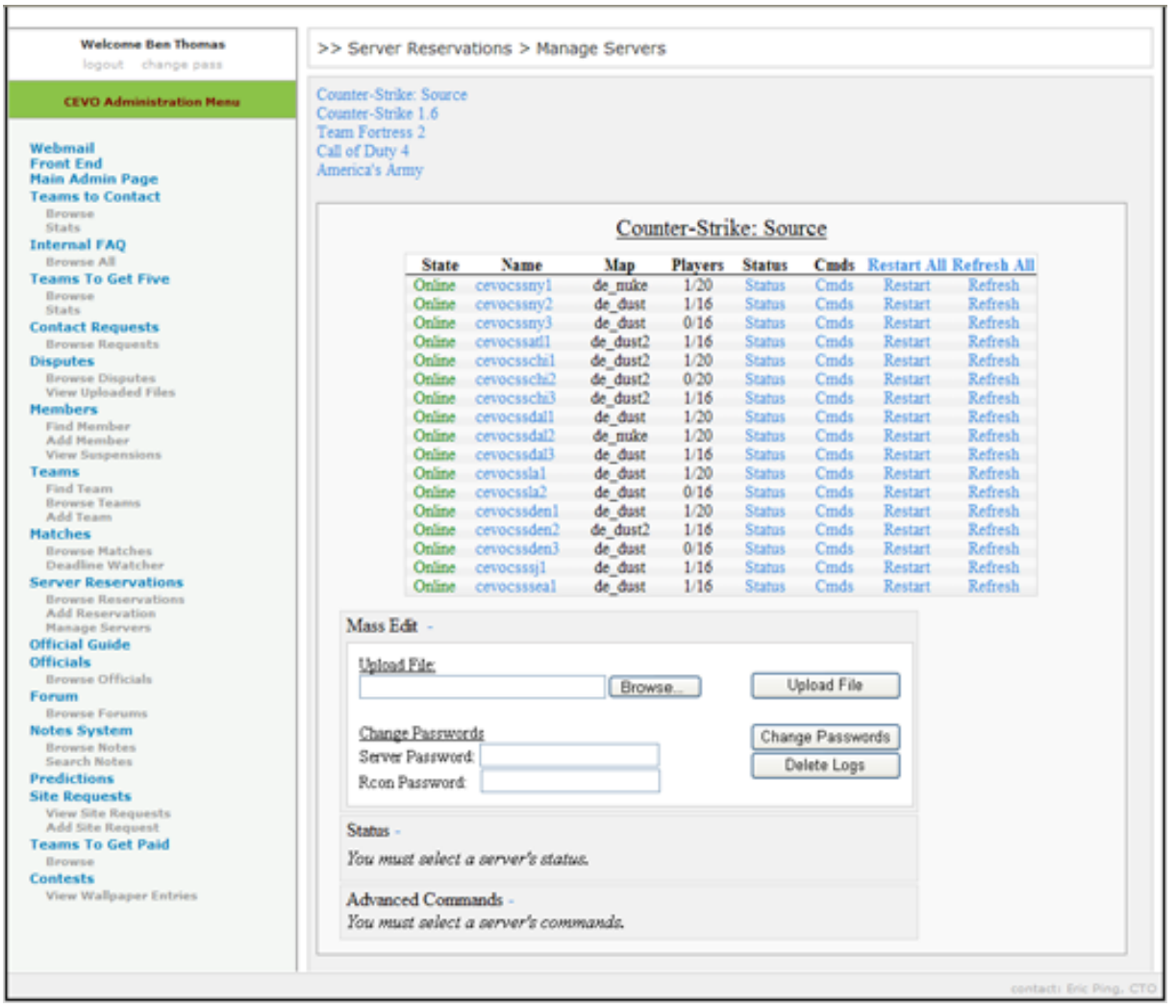

Figure 5 – CEVO server management

The CEVO system excels when it comes to league style competitions that are held over an extended time frame as the system allows competitors to schedule their own matches via the front end. This is a feature that although exists at GotGames will be omitted from this work in order to concentrate on automating single and double elimination competitions where an aggressive timeframe is important. Figure 6 demonstrates a listing of matches on the CEVO system, each match can be edited by an administrator however they are originally scheduled by competitors themselves.

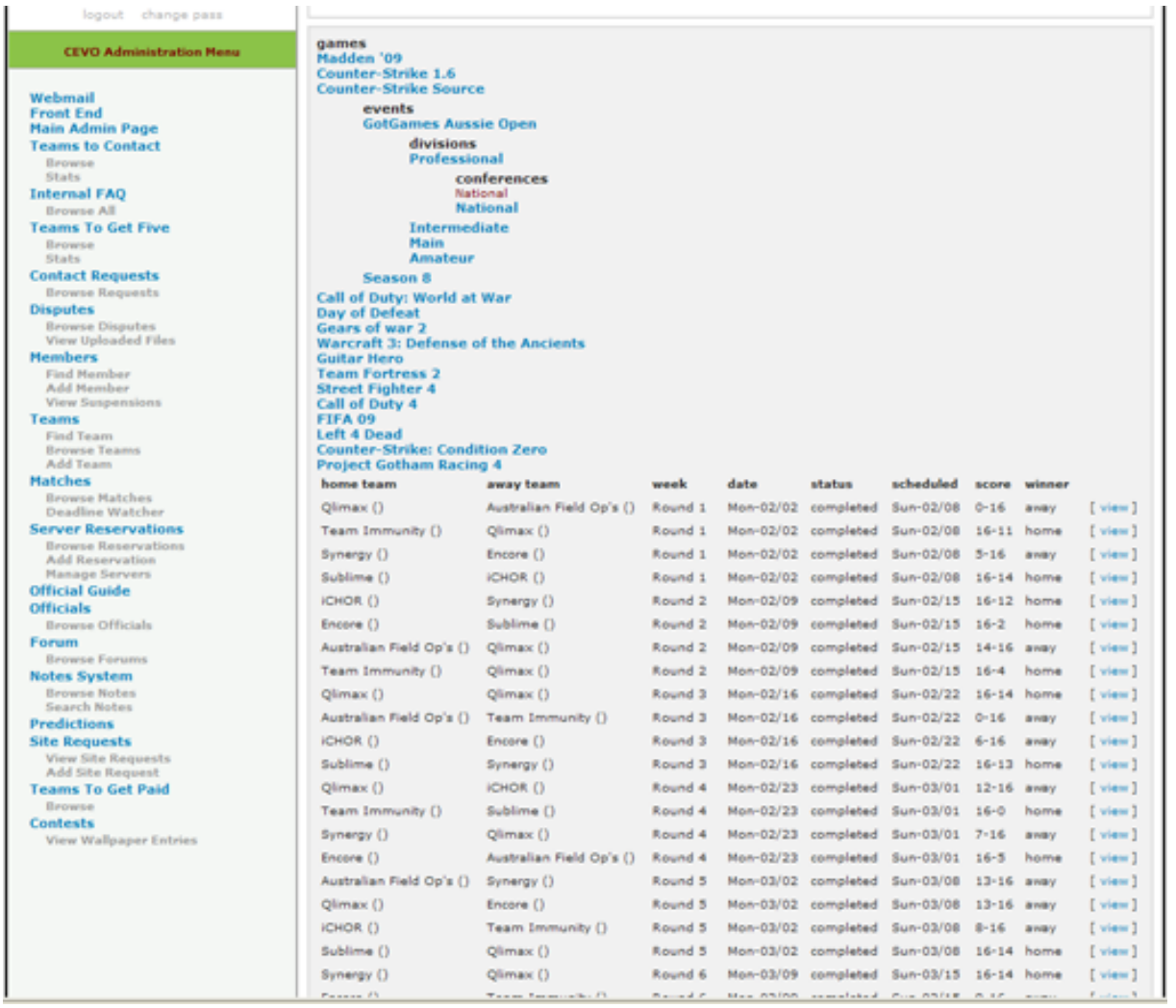

Figure 6 – CEVO match listing

The CEVO client as seen below is considered to be the most reliable identification verification and anti-cheat tool available today(John Sidey 2007). However once again the client is linked with the CEVO system and only tournaments run via CEVO can utilize the client in any form. Figure 7 shows the main display of the CEVO client as scene on the competitor's PC.

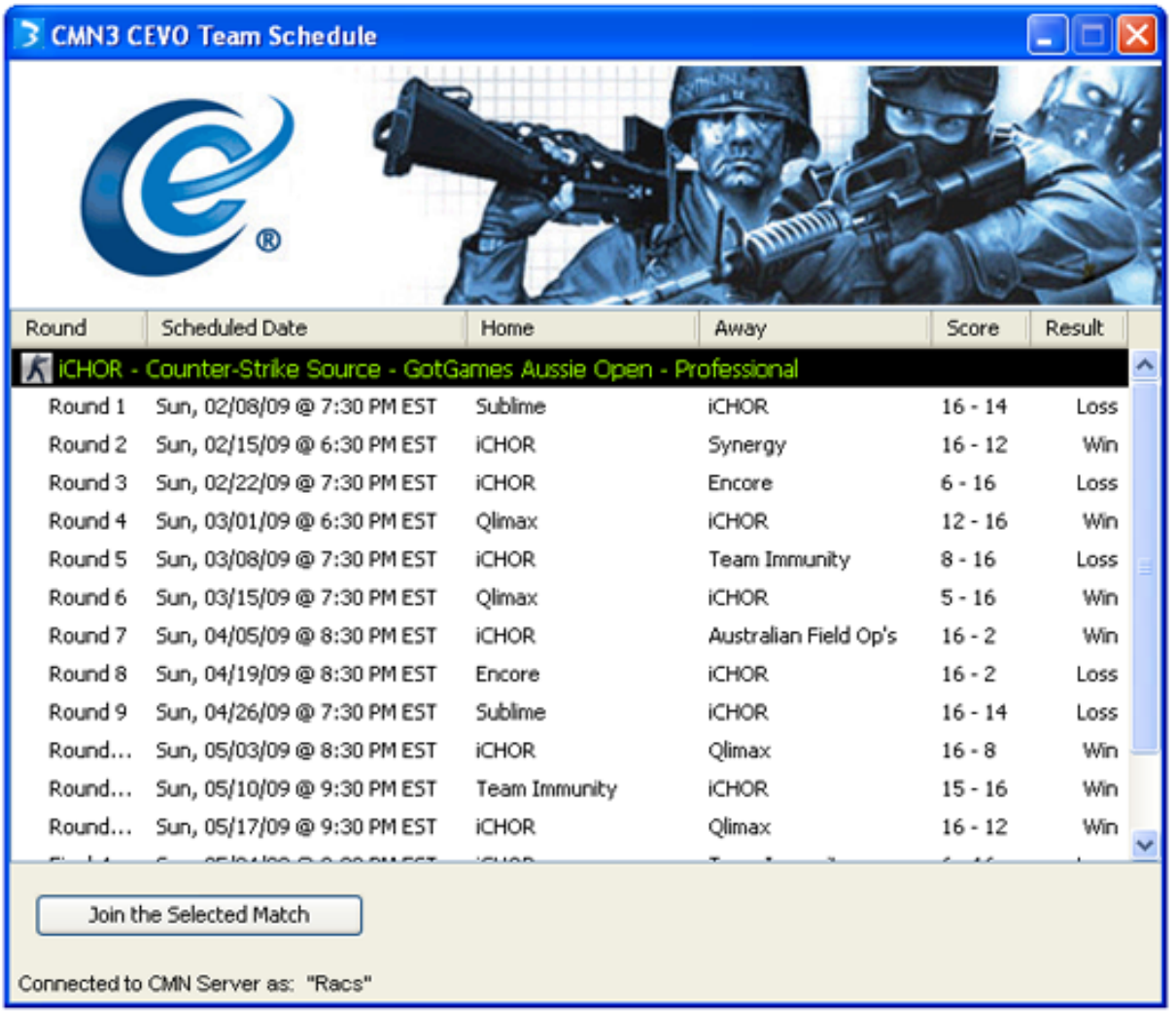

Figure 7 – CEVO client

The CEVO system is a fully function tournament management software solution that does automate a majority of work involved in scheduling and server management however it is not suitable for GotGames deployment because:

- The system does not allow for the addition of Servers that are not owned and operated by CEVO.
- The system is not capable of managing Source TV relays.
- Functionality to display information relating to upcoming matches and spectator details is lacking.
- The system is implemented as a single entity and as such all the modules and features are hard coded for CEVO requirements and into existing CEVO infrastructure.
- The CEVO client requires the operation of a Client Server which is based in the United States and results in a magnitude of lag issues for Australian competitors.

2.5.2 Bracket Maker

Aside from corporate systems like the CEVO system mentioned above there are a number of smaller assets available to assist in the management of online tournaments. Although tools for server management do not exist outside the corporate realm there are some tools that assist in scheduling and sharing match information in a human readable format.

One such tool is available online and is called BracketMarker, although BracketMaker can automate brackets it requires the user to enter all manual data for each and every bracket created. Another downfall of the Bracket Maker software is that details for every single match must be manually entered and updated by administrators.

Figure 8 – Bracket Maker example

2.5 Information Feeds and Protocols

This section examines the options available for feeding tournament data including, match schedules, results and spectator information to external reporting websites. In particular this section focuses on three available options for sharing this information; JSON, XML and direct database access.

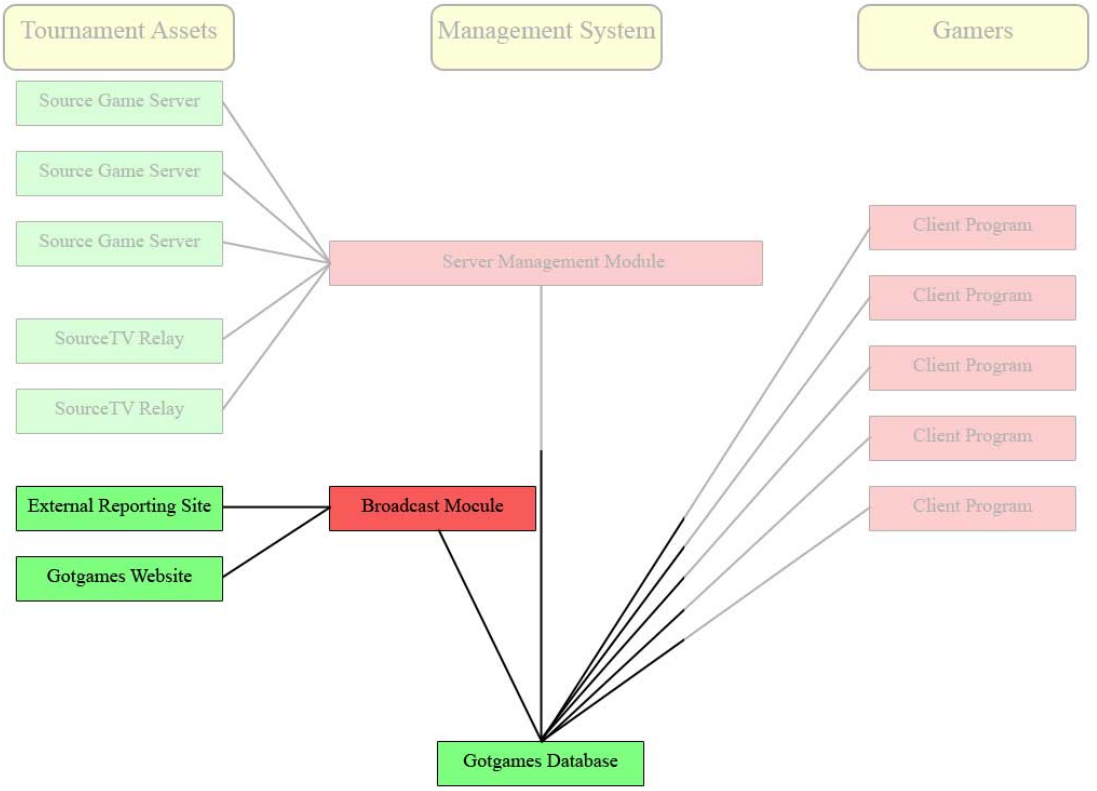

Figure 9 - Broadcast module

2.5.1 JSON

JavaScript Object Notation is a lightweight data interchange format that is text based and human readable and is used for describing simple data structures referred to as objects. One area where JSON excels is in Ajax-style web applications where web based

applications use lightweight out of band calls to the web server instead of full page refreshes. Since it is primarily a data format, JSON is not limited to just Ajax web applications, and can be used in virtually any scenario where applications need to exchange or store structured information as text.

Array Support	**Native support for arrays.**
Size	Most of the space is consumed by the represented data because the syntax is very strict.
Namespaces	No support for namespaces, this is overcome by nesting objects.
Object Support	Native support for objects.
Data Types	Structured data can be represented through arrays and objects while support for scalar data types is present.
Null Support	Native support for the null identifier.
Formatting	Concise and simple due to strict syntax requirements.
JavaScript Parsing	Can be easily parsed by JavaScript using eval.
Comments	Comments are not supported; however the format is extremely easy to read.

Table 1 - JSON Characteristics

GotGames currently make heavy use of Ajax-style web applications, Java and JavaScript and for this reason it is important to consider JSON as a viable data format and exchange protocol.

2.5.2 XML

Extensible Markup Language is a markup language for documents containing structured information. Structured information contains both content and some indication of what role that content plays. An example would be content for a title section of a document would play a different role to content that was designed for a footnote. Almost all documents have some structure. A markup language is a mechanism to identify structures

in a document. The XML specification defines a standard way to add markup to documents.

XML is a text based format and was designed specifically to carry data and not to display data. Although similar in structure to HTML, XML does not provide document formatting but rather only contains document data. Tags in XML are not predefined and must be defined; these tags are then read and formatted through the use of another application.

Array Support	**Arrays support is not native and array must be expressed by conventions.**
Size	Documents are often lengthy.
Namespaces	Supports namespaces.
Object Support	Objects must be expressed through conventions.
Data Types	No data type notation.
Null Support	No native support for null identifier.
Formatting	Can become very complex and requires careful planning.
JavaScript Parsing	Requires an XML DOM implementation and additional application code to map text back into JavaScript objects.
Comments	Native support for comments.

Table 2 - XML Characteristics

The above table can be compared to Table 1 on page 26 in order to signify the main differences between XML and JSON. Both languages offer different features for sharing formatted data between application and across networks. One major advantage of XML over JSON is the large number of mature tools available for use with XML as the protocol is over 10 years old.

2.5.3 Direct Database Access

As the name implies this method requires outside applications to have direct access to data stored in database tables (SQL in the case of GotGames). Although by far the easiest and fastest method of sharing data, allowing uncontrolled access to a database results in a number of security and performance concerns. Unregulated database access can result in:

- Theft of sensitive or valuable data.
- Performance losses due to an unknown number of additional queries.
- Intentional disruption of services.
- Daisy-chaining of unregulated access through authentication progression.

Obviously the speed and ease of implementation that are offered by direct database access cannot account for the obvious security and performance risks associated. Even allowing limited direct database access can result in massive performance losses that will affect all processes on that server. For the reasons mentioned above direct database access will not be considered as a viable option in Section 3.2 where protocol decisions are discussed.

2.6 Summary

Although brief the information provided in Chapter 2 is required in order to set a solid base for design and implementation decisions. Basic knowledge of the software, systems, protocols and history provide a solid foundation for informed decision making in later Sections. To summarize:

- E-sports is growing locally especially in an online environment.
- Online e-Sports tournaments require automated management systems to alleviate massive amounts of manual work handled by GotGames staff.
- User identification and authentication is a major hurdle facing online competitions.
- GotGames has made available a number of assets that will be required in order to complete this work.
- Valve is responsible for the creation of all game and game platform software and both are automatically updates frequently.
- Steam is the platform used to launch all Source based games.
- Steam has built in authentication for users and each account is linked to a unique Steam ID.
- The game engine that this project has selected to concentrate on is the Source Engine.
- Aside from the game client Valve have also developed stand-alone or dedicated Server and Spectator services for the Source Engine.
- RCON is a remote administration tool that can be used to manipulate Source Servers and Source TV services.
- Tournament formats that this project is concerned with are single and double elimination.
- There are existing solutions for tournament management but they are not scalable or are privately owned and customized for specific tasks.
- Sharing tournament data is a feature lacking in all available tools and is essential to boost interest and spectator numbers.
- There are a number of formats to consider for data sharing including JSON, XML and direct database access.

CHAPTER 3

MODULE DESIGN AND DEVELOPMENT

3.1 Introduction

This chapter contains in-depth discussion regarding module design and development. Each module is contained in a section as well as a brief overview of the design philosophy and existing infrastructure that required interfacing. Some modules have been broken into smaller sections in order to adequately cover the complexity of the system.

3.2 System Overview

This section contains information on the overall system and interaction between modules as well as examining the existing infrastructure at GotGames. Discussion about design choices including:

- Coding languages;
- Module breakdowns;
- Scripting Options;
- Operating environments;
- Resource requirements;

3.2.1 GotGames Resources and Infrastructure

The existing infrastructure must be carefully considered when interfacing new modules. GotGames has a number of machines serving as game servers, web servers and database servers. These servers predominantly operate in the Linux environment however there are several machines that are still dependant on Microsoft Windows Server including the machine responsible for web serving.

From a software perspective GotGames has a number of useful modules in operation including:

- VB Forums that are used to manage user information and details.
- A team system that is used to link users to particular teams.
- A recent server booking application that requires the ability to interface with Source Servers.
- GotGames Live system that allows users to communicate, advertise and play games in groups or privately through the use of JAVA applets.

All of these software packages interact on some level and rely on heavy custom coding and scripting. It is important to note that some of these systems rely heavily on Ajax-style web applications that share information through the use of JSON. Most of the modules currently in use and development with the GotGames system are PHP, SQL, JAVA or JavaScript based.

3.2.1.1 GotGames Database Schema

GotGames operate a single database responsible for storing information from a number of different application and services. Because of the importance of database integrity on current services it was appropriate that clones of existing tables be created for testing and data manipulation. A detailed report on the tables used in development can be found in Appendix I. Tables prefixed TMS are clones of existing GotGames tables while the rest are actual tables as they exist within the GotGames database.

It is important to note that the additional load created by module queries is negligible when compared against the daily load place on GotGames servers. Therefore optimization to reduce query count was not considered in the coding of tournament management modules.

3.2.1.2 User Interface Requirements

Existing interfaces allow GotGames users to modify and access most of the relevant data for team and player management. However in order to keep impact on current operations as low as possible when creating tournament management modules basic interfaces where created to facilitate the addition and editing of some data. These interfaces are constructed in PHP and can be located in Appendix E.

3.2.2 Language and Protocol Choices

The primary goal of language and protocol selection is to make interfacing and creating of modules as simple as possible while maintaining a robust function set to achieve the desired results. It is obvious that selecting languages already in use at GotGames would make integration a simpler task. Table 3 shows the languages and protocols considered and why each was chosen or disregarded:

Language/ Protocol	Pros	Cons	Final Decision
Perl	• Simple • Powerful • Not platform specific	• Outdated • Hard to understand • Limited toolset • Limited knowledge • Hard to integrate	Rejected Perl as a viable option because PHP offers a richer environment with better future support.
PHP 	• Extremely simple • Well structured • Abundant support • Rich feature set • Large toolset	• Open source • Poor error handling	PHP was chosen as the scripting language due it the available toolsets and support.
C/C++	• Powerful • Popular • Numerous libraries • Compiled language	• Steep learning curve • Platform specific code • Networking not standardized	C++ was rejected mainly due its poor handling of network and socket code from platform to platform.
JAVA 	• Cross Platform Support • Large libraries • Open source • Relatively easy	• Requires JVM • Limited Vision outside of JVM	Java was chosen simply because it handles sockets with ease and is cross-platform.
JSON 	• Well Structured • Works well with JAVA • Lightweight • Simple • Object support • Datatype support	• No namespace	JSON is the obvious choice for a system already reliant on JAVA and JavaScript.
XML	• Robust • Mature • Comments • Human readable	• No object support • Size • No datatype support	XML was rejected because JSON was the better option for integration.

Table 3 - Language choices

3.3 RCON Interface Module

This section will examine the interface required for Source Server and Source TV manipulation. Topics covered include the RCON protocol and a discussion of the final working module.

3.3.1 RCON Protocol

The first step to designing and developing an interface between Source Servers/Source TV and the management system is a comprehensive understanding of the RCON protocol. The RCON protocol defines how information is sent to and received from the servers and limited information is available outside of the Valve offices. The protocol itself is based around command and response packets that are encapsulated in a TCP/IP stream.

3.3.2 Sending Packets

In order to authenticate the RCON password the first packet sent to any Source Server must contain a SERVERDATA_AUTH command. If this is not the case and a command is sent instead of the auth then the response from the server will indicate a SERVERDATA_AUTH_RESPONSE where the failure flag is set.
Command packets sent to the server have the following structure:

- packet size (int) - the number of bytes from the start of the requestid to the end of string2 (including the null byte). It must be at least 10.
- request id (int)
- SERVERDATA_EXECCOMMAND / SERVERDATA_AUTH (int) SERVERDATA_AUTH is currently 3 SERVERDATA_EXECCOMMAND is currently 2
- string1 (is the command to run).
- string2 must be null ("");

Note that the integer values for the AUTH and EXECCOMMAND bytes are preset and have been known to change with major updates.

3.3.3 Receiving Packets

Before we look at the structure of the received packet it is important to note that Source Servers send one junk packet during the authentication step, before they respond with the correct authentication response.

The response packet is much the same as the command packet:

- packet size (int)
- request id (int)
- command response (int) - valid command responses being:
  SERVERDATA_RESPONSE_VALUE = 0
  or SERVERDATA_AUTH_RESPONSE = 2
- string1 (null delimited string)
- string2 (null delimited string)

In order to interpret responses from the server the following table may be used:

Sent Command	Value	Return
SERVERDATA_AUTH	Correct Password	Mirrored ID
SERVERDATA_AUTH	Wrong Password	-1
SERVERDATA_EXECCOMMAND	Any	-1

Table 4 - SERVERDATA_AUTH_RESPONSE codes

Sent Command	Value	Return
SERVERDATA_EXECCOMMAND	Command String	String

Table 5 - SERVERDATA_RESPONSE_VALUE codes

A single SERVERDATA_EXECCOMMAND command may result in multiple SERVERDATA_RESPONSE_VALUE response packets due to that fact that the response string (string1) has a maximum length of 4096 bytes.

3.3.4 RCON Design and Development

As already discussed in Section 3.2 were possible all coding will be completed in JAVA while were possible all scripting will be completed in PHP. This module is required to execute RCON commands (see Appendix B**)** on remote Source Servers and Source TV Services. For this reason this module will be interfaced directly between the Servers /TVs and the server management module as shown in figure 10:

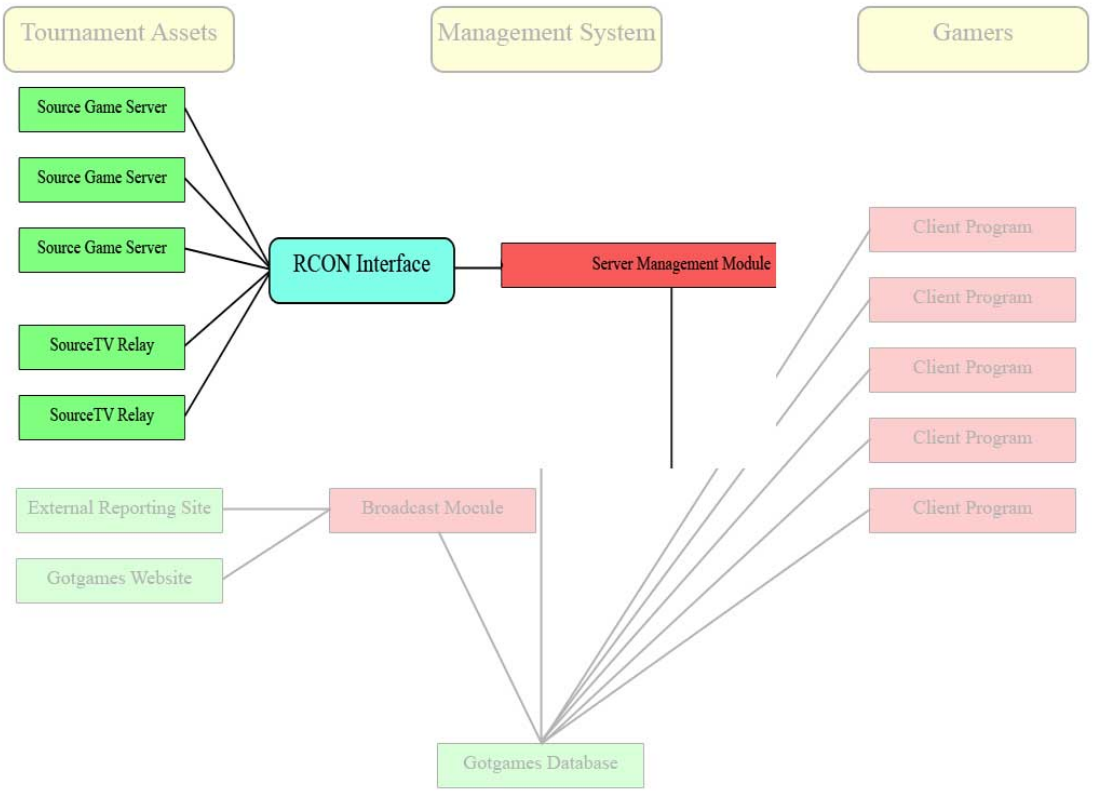

Figure 10 – RCON interface

From the above Figure it is clear that neither the competitor nor the administrator of the tournament has any interaction with the RCON Interface and therefore the module will not require a GUI but rather an API. Therefore the simplest and most robust way to design the module was a simple executable with a command line capable of taking the RCON arguments. The arguments required for the RCON interface are:

- Server/TV IP address.
- Server/TV Port.
- RCON Password.
- RCON Command.

An executable can be easily called from the PHP script and values can be passed to the command line during the same call.

To begin with the module requires the ability to construct RCON packets which can be easily achieved in JAVA using the ByteBuffer object. The source code containing detailed comments can be found in Appendix F. The code is a simple implementation of JAVA socket functions and requires no further discussion. It is important to note that presently there is no error handling for the custom exceptions BadRcon and ResponseEmpty and on successful execution of the RCON command the executable exits providing no return data. Finally the format for calling RCON commands from the command line is:

java –jar rconinter.jar **Server_IP_Address Server_Port RCON_Password Command**

3.4 Client Software

Covered in this section is the design and implementation of the client software and the associated Chat Server. Segments of the server code are altered from an original JAVA chat server designed by IBM. The client and server software will be implemented in JAVA as per Section 3.2.2 Language and Protocol Choices.

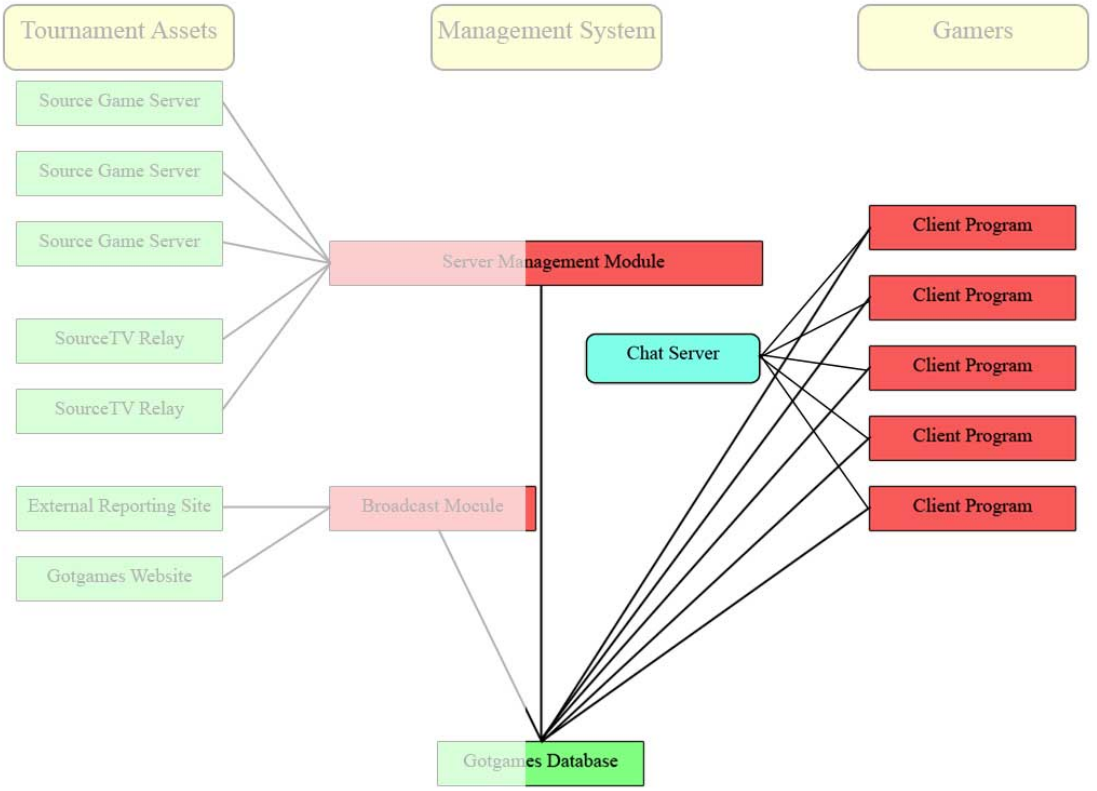

Figure 11 – Client/Server module

3.4.1 Client Functions

Chapter 2 covered the importance of verifying player's identity throughout the course of an online tournament and the Client is designed to fulfill those needs. In order to authenticate players the client software checks:

- Verify username and password with those stored in the GotGames database and used for the players GotGames account.
- Verify Steam Platform is installed.
- Verify local Steam ID with that saved in the GotGames database.

3.4.1.1 GotGames Authentication and Scheduling Information

The first identity check that the Client software is required to do is to verify that the player's login details match those recorded in the GotGames SQL database. In order to authenticate this information the client is required to access the GotGames database and retrieve details about the user attempting to login. The SQL schema present at GotGames was covered in Section 3.2.1.1 and all the relevant user information is stored in the table vb_user.

Accessing this data is a trivial task of connecting to the SQL database using the mySQL database driver and retrieving the desired information from vb_user. A separate JAVA class called TMSLocalUser was created to handle the collection of all relevant information from the GotGames database, the source for which can be found in Appendix H. The following segment of code demonstrates execution of a single SQL query in JAVA:

```
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM vb_user WHERE username =
'"+Username+"';");
        rs.first();
```

```
GGSalt = rs.getString("salt");
```

The class stores all the gathered details in public variables and uses a simple error toggle system to indicate that an error has occurred by simply setting the IsError flag. Once the information is retrieved from the GotGames database the passwords can be compared.

It is important to note that the system used by GotGames encrypts the passwords using MD5 and a Salt value. The MD5 system is used to hash the original password then the salt is added and the process is repeated. In order to compare passwords the client encrypts the password entered by the user in the same method using the salt value retrieved from the GotGames database. The following segment of code is responsible for encoding of the entered password:

```
//lets MD5 has the enter password
try {
   MessageDigest md5 = MessageDigest.getInstance("MD5");
   md5.update(Password.getBytes());
   BigInteger hash = new BigInteger(1, md5.digest());
   hashword = hash.toString(16);
   StringBuffer buffer = new StringBuffer(hashword);
   while (buffer.length() < 32) {
      buffer.insert(0, '0');
   }
   firststage = buffer.toString(); //first stage is complete
   } catch (NoSuchAlgorithmException nsae) {
   }
hashword = null; //reset hashword
Password = firststage+GGSalt; // add the salt retrieved from the GG database
// MD5 hash the new value so we can compare the outcome.
try {
   MessageDigest md5 = MessageDigest.getInstance("MD5");
   md5.update(Password.getBytes());
   BigInteger hash = new BigInteger(1, md5.digest());
   hashword = hash.toString(16);
   StringBuffer buffer = new StringBuffer(hashword);
```

```
        while (buffer.length() < 32) {
            buffer.insert(0, '0');
        }
        hashword = buffer.toString();
    } catch (NoSuchAlgorithmException nsae) {
    }
```

Encoding the password locally on the users machine means that only the hashed password sequence is never transmitted over the socket. Once the client has verified a users login details the scheduling information for the next match is retrieved from the database and transmitted to the client for display to the user.

3.4.1.2 Verifying Install and Local Data

TMSLocalUser is responsible for gathering all local information from competitors PC. As with most modern windows application the Steam Platform uses the Windows registry to store details about installation including the install path. By checking that the install path exists in the registry key 'SOFTWARE\\Valve\\Steam\\InstallPath' it is possible to verify that Steam is installed and also recovery the location of the steam log. Once the location of the log is found local data including Steam ID and Steam Account Name can be read from the log. The function GetLocalInfo in TMSLocalUser (Appendix H**)** is responsible for retrieving this information and storing it in class variables.

Once the information from the log is recovered it is simply compared to the Steam ID already obtained from the GotGames database as explained in Section 3.4.1.1 GotGames Authentication

3.4.1.3 Client Chat

A very simple global chat system has been added to the client software to enable users logged into the client to communicate with one another. The chat system required the creation of a threaded chat server that will be discussed in Section 3.4.1.4 Chat Server.

The source code for the chat system can be found as part of the TMSClientFinalMain class found in Appendix H.

The chat service listens for user input and calls processMessage() when the user input event is triggered. proccessMessage() is parsed the string the user entered and writes that data to the server before clearing the text input field. The messages are parsed with the username of the sender in order to keep track of message origins.

The chat segment of the client software is also listening for data to be sent from the server. The TMSClientFinalMain constructor creates a thread as its last task; the purpose of this thread is to read incoming messages from the server. This thread displays each incoming message to the text are then goes back to waiting for more incoming messages.

3.4.1.4 Chat Server

The chat server is a standalone JAVA application in the form of a single process running on a server machine. The code for the Chat Server can be found in Appendix G and is contained with the Server class. The server listens on a port parsed on the command line and uses sockets to communicate with each client.

The Chat Server is a threaded application in the sense that each new socket connection is handled by a separate thread. The source code for the thread can be found in Appendix G. This thread simply listens for incoming data from a client then once data is received broadcasts that data to every other connected client. It is important to note that server side time stamps are used for the server status and the log these form can be used to identify when each user logs in and out of the system. Finally it is the job of ServerThread to inform the main Server class of closing connections to avoid memory leaks and wasting time attempting to communicate to closed sockets.

3.4.2 Automated Scheduling and Bracket Functions

Taking a closer look at the server management model will allow us to break it into three distinctive sections:

- Auto-scheduling module.
- Bracket building module.
- Server booking system.

GotGames has recently implemented a SQL based server booking module and coupled with the RCON interface covered in Section 3.3.4 RCON Design and Development is capable of booking servers by means of command line parsing to an external process. This section will therefore focus on the Auto-Scheduling and Bracket Building features of the tournament management system.

Figure 12 - Scheduling and bracket building

Section 3.3 and 3.4.1 covered the modules and services that have been developed in the JAVA language. As figure 12 indicates the scheduling and bracket building services communicate directly with the database and therefore are more suited to a scripted language. The following modules have been developed purely in PHP and are concerned solely with the manipulation and formatting of database values.

3.4.2.1 Automated Scheduling Module

The auto-scheduler (source code can be found in Appendix D) was designed to automate the process of setting individual matches for each tournament. The auto-scheduler takes into account the number of available Source Servers and Source TV Relays and assigns each match the following:

- Time.
- Date.
- Source Server.
- Source TV if available.
- Server Password.
- Bracket Position.
- Team information is available otherwise information on the parent match.

The auto-scheduler requires a start date and time to be entered along with the timeframe for each round. It is assumed that only one round of each bracket is to be played per timeframe and matches are calculated to be completed in the shortest possible timeframe. This results in the maximum number of matches running concurrently on different Source Servers.

The auto-scheduler processes each round of the competition and saves the calculated matches. Scheduling of a single round involves:

1. Calculating total number of matches

2. Calculate the total number of matches that can be played at once using the number of available Source Servers.
3. Retrieving the data pertaining to the Source Servers and Source TV Relays available.
4. Taking the first seeded team and last seeded team and creating a match
5. Removing these teams from the teams array.
6. Assign a server to this match
7. Removing the assigned Source Server from the server array
8. Assign a server password to this match
9. If available assign a Source TV Relay to this match
10. Removing the assigned Source TV Relays from the stv array
11. Save the match Information
12. Repeat from step 3 until there are no teams left in the teams array.

By assigning the available Source TV Relays in this manner it is effectively guaranteed that the highest seeded teams will have spectator facilities. All Match information is saved in the tms_matches table and can be parsed directly to the existing booking system. For each match the data field position is used to store that matches position on a virtual bracket to facilitate bracket building as covered in the next section.

3.4.2.2 Bracket Module

The Bracket Module is responsible for formatting and displaying a bracket in human readable format. The source code for the Bracket Module is located in Appendix C. Figure 3 on page 16 depicts a simplified version of the common bracket, this module builds on that bracket by providing the date, time, teams and spectator information for each match.

In order to display the bracket HTML tables have been used as the layout engine. HTML is designed to display data from left to right then top to bottom however the brackets have been constructed from top to bottom then left to right. Using HTML to layout the bracket

is comparative to taking a pyramid that was built from the ground up and turning it on its side.

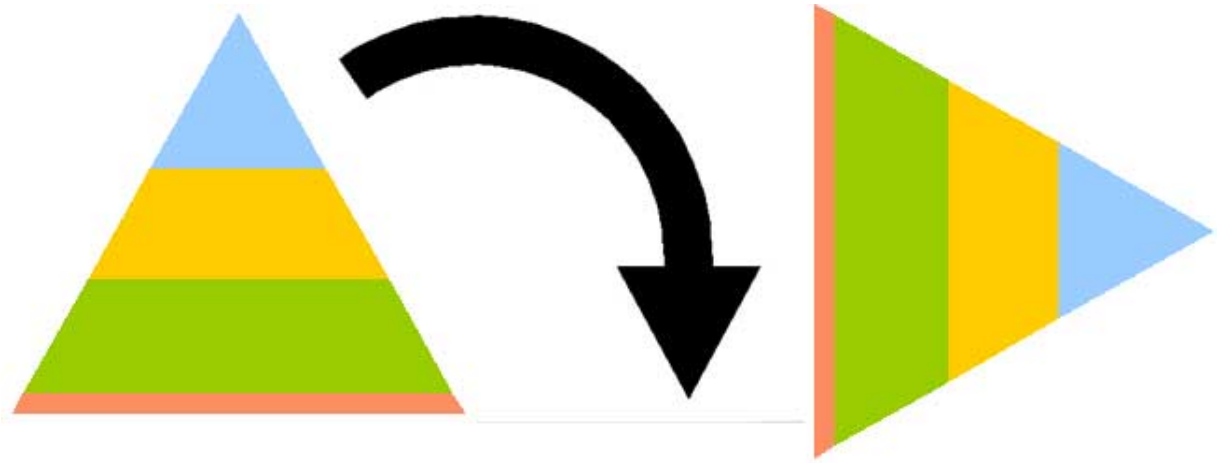

Figure 13 – Bracket layout

To account for this rotation in construction the Bracket Module first creates an array for each table entry. This array holds data that in turn is used to decide how to fill each table cell when it comes to displaying the bracket. Using an array to represent the overall picture allows data to be filled from top to bottom and the complete array to be drawn from right to left.

The following code segment is used to create the array representing the complete bracket:

```
for($i = 1; $i <= $total_rounds; $i++) {
                $round_matches = $this->getmatches($i);
                $matches = pow(2,$total_rounds-$i); //calc how many matches for this round
                $interval = pow(2,$i); // calc the interval for layout spacing
                $offset = pow(2,$i-1);
                        for($c = 1; $c <= ($this->tournament_size*2)+1; $c++) {
                                if ($c < $offset) { // blank space
                                        $col[$i][$c] = 0;
                                } elseif ($c > (($this->tournament_size*2)+1)-$offset) {
                                $col[$i][$c] = 0;
                                } elseif ($c==$offset) {
```

```
                    if ($i==$total_rounds) { //No match but tournament
                    $col[$i][$c] = "Champion";
                    }else { //print a team here
                    $tmp_array=array_shift($round_matches);
                    $col[$i][$c] = $tmp_array['print_team'];
                    }
                } elseif ((($c-$offset) % $interval) == 0) { //print a team here
                    $tmp_array=array_shift($round_matches);
                    $col[$i][$c] = $tmp_array['print_team'];

                } elseif ($c==($offset*2)) { //print match here
                    $col[$i][$c] = "Match".$tmp_array['id'];
                } elseif ((($c-$offset*2) % ($interval*2)) == 0) { //print match
                    $col[$i][$c] = "Match".$tmp_array['id'];
                } else {
                    $col[$i][$c] = 0; // blank space
                }
            }
            print("<br>");
    }
```

In order to locate the edges of the pyramid and match cells the code users an offset and interval that are calculated for each column. The offset is used to find the edge of the pyramid and is defined as:

$$offset = 2^{current\,round-1}$$

The interval is used to space out each match between the pyramid edges and is defined by:

$$interval = 2^{current\,round}$$

Like the auto-scheduler the bracket maker is coded in PHP and interacts directly with data stored in the GotGames database.

3.5 Summary

The information covered in Chapter 3 provided insight into the design and implementation decisions made during module construction. The Chapter covers the existing infrastructure and resources available at GotGames as well as the basic fundamentals behind the operation of each module. To summarize:

- JAVA and PHP were the main languages used in the construction of modules.
- The broadcast module would benefit from the use of JSON.
- GotGames has existing systems for server booking however implementation of the RCON module was required to provide support for Source Servers.
- The RCON module is a command line tool that communicates to servers through sockets.
- The Client software authenticates a user by checking the GotGames login and Steam ID details.
- Local Steam ID details are obtained through the use of the windows registry to locate the Steam log file where this information is stored.
- Chat functionality has been added to the Client software and is threaded.
- A Chat server was developed and the logs from the Chat server can be used to verify a users online time.
- The Auto-Scheduler and Bracket modules interact directly with the database and are written in PHP.
- Simple input scripts have been constructed to allow manipulation of tournament data.

CHAPTER 4

MODULE TESTING

4.1 Introduction

This chapter discusses the output obtained from sample executions of each module. Discussion of current bugs and indentifying limitations of each module will also be covered. The chapter will be broken into individual sections so that output from all code can be examined.

4.2 Test Procedures

Because the work involved in this project is concerned with the development of separate modules and not a complete tournament management system each module was tested with a variety of dummy inputs. The following sections supply a sample test output along with brief discussion of the overall test results for that module.

4.2.1 RCON Interface

To test the RCON Interface a number of RCON commands were issued to the command line tool and the output read from the server logs. Below is a sample test case:

Input:

```
I:\Documents and Settings\Racs\My Documents\NetBeansProjects\rconinter\dist>java
 -jar rconinter.jar 118.127.16.173 31007 sH!z "say no"

I:\Documents and Settings\Racs\My Documents\NetBeansProjects\rconinter\dist>java
 -jar rconinter.jar 118.127.16.173 31007 sH!z "changelevel de_nuke"
```

Output:

```
08:06:07 L 01/03/2010 - 09:06:09: rcon from "220.253.166.147:2118": command "say no"
08:06:16 L 01/03/2010 - 09:06:17: rcon from "220.253.166.147:2119": command "changelevel de_nuke"
```

The command line RCON interface worked as expected in 100% of test cases. Every RCON command was received and executed by the server and the lack of error handling meant that invalid commands were not processed.

4.2.2 Client and Chat Server

Testing of the Client and Chat Server was done on both a local machine and within the internet environment. A sample test indicative of those performed follows:

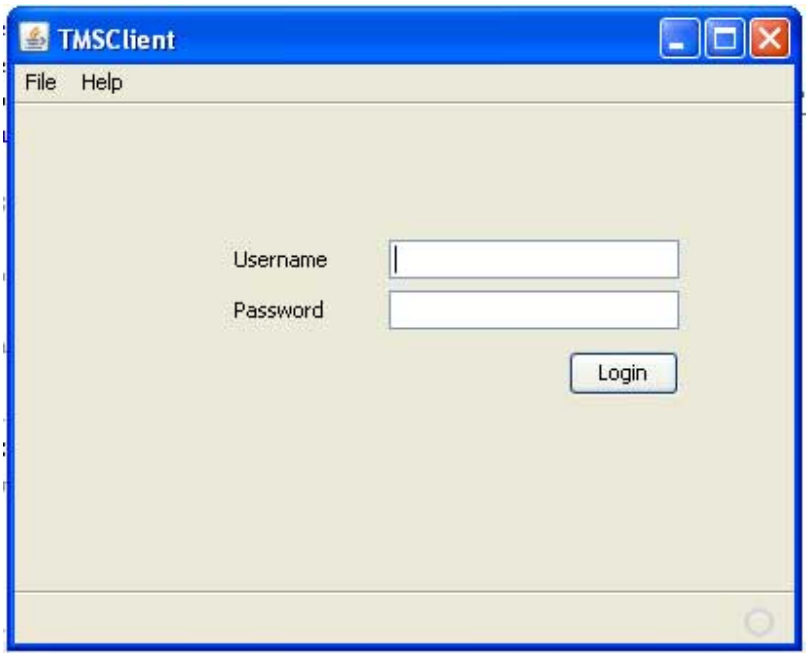

Figure 14 – Client login screen

The client was logged in using valid GotGames account details and the message ‘Hello’ was typed and sent. Another client was then run on the same machine and logged in to the same GotGames account, however the message ‘Goodbye’ was sent by the second client.

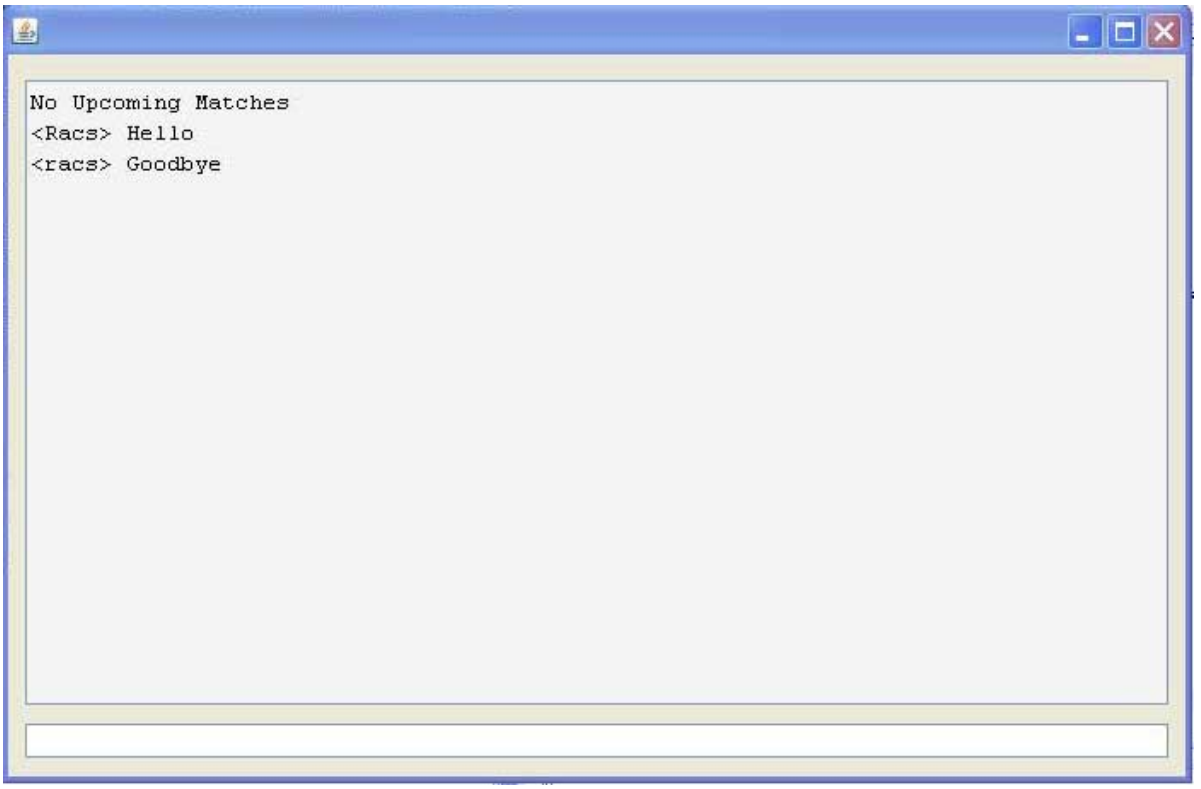

Figure 15 – The first client

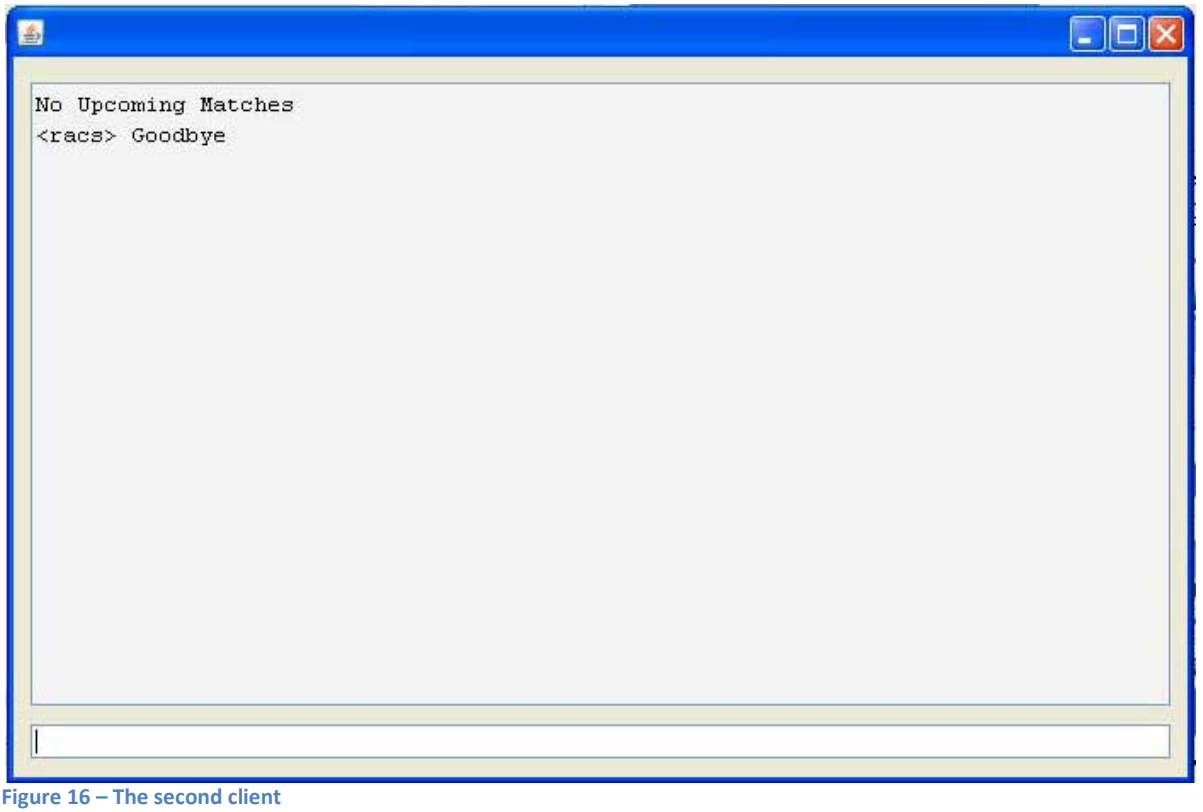

Figure 16 – The second client

Both clients where then exited and the results from the chat server logs examined:

```
>java -jar server.jar 10000
Listening on ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=10000]
2010-01-03 08:17:12 Connection from Socket[addr=/127.0.0.1,port=2152,localport=1
0000]
2010-01-03 08:17:18 Sending <Racs> Hello
2010-01-03 08:17:48 Connection from Socket[addr=/127.0.0.1,port=2154,localport=1
0000]
2010-01-03 08:17:54 Sending <racs> Goodbye
java.net.SocketException: Connection reset
        at java.net.SocketInputStream.read(Unknown Source)
        at java.net.SocketInputStream.read(Unknown Source)
        at java.io.DataInputStream.readUnsignedShort(Unknown Source)
        at java.io.DataInputStream.readUTF(Unknown Source)
        at java.io.DataInputStream.readUTF(Unknown Source)
        at ServerThread.run(ServerThread.java:39)
2010-01-03 08:19:13 Removing connection to Socket[addr=/127.0.0.1,port=2152,loca
lport=10000]
java.net.SocketException: Connection reset
        at java.net.SocketInputStream.read(Unknown Source)
        at java.net.SocketInputStream.read(Unknown Source)
        at java.io.DataInputStream.readUnsignedShort(Unknown Source)
        at java.io.DataInputStream.readUTF(Unknown Source)
        at java.io.DataInputStream.readUTF(Unknown Source)
        at ServerThread.run(ServerThread.java:39)
2010-01-03 08:19:20 Removing connection to Socket[addr=/127.0.0.1,port=2154,loca
lport=10000]
```

As expected both the client and the server operated as per the design goals. It is important to note that during the tests the client failed to load and displayed the appropriate error message on occurrences of:

- Steam not being installed.
- Incorrect login details supplied.

- Steam ID mismatches.

Testing identified a situation where if the Chat Server was not running the Client failed to notify the user and did not allow messages to be entered. Possible improvements to the chat system and the client server model are covered in Section 5.3.

4.2.3 Auto-Scheduler

Tournament data was required to test the Auto-Scheduler and therefore new test tournament data was added to the database using the basic custom PHP scripts that can be found in Appendix E. The input screen for create.php is shown below, although the scripts teamselect.php and addsever.php were also used to create test data they will be omitted in the interest of brevity:

Create.php

Figure 17 - Createserver.php display

Once all the required test data had been entered the Auto-Scheduler was run. The test data consisted of a 16 team single elimination competition with 2 Source Servers and 1 Source TV Relay available for use. Rounds were set to take place each night and match lengths were set to one hour. The resulting information in tms_matches exactly matched the expected outcome. Figure 18 shows the result of the Bracket Builder on this test data.

Extensive testing of the single elimination format yielded perfect results. However the double elimination format is incomplete at this stage and contains a multitude of bugs causing incorrect data to be saved.

4.2.4 Bracket Builder

Data from the test case discussed in Section 4.2.3 Auto-Scheduler, was used to construct a meaningful competition bracket. Brackets can only be constructed once the Auto-Scheduler has completed calculating match details for every match in the competition. Using the output from the above Auto-Scheduler test yielded this bracket:

Figure 18 - Bracket Builder output

In all test cases the Bracket Builder performed as expected with no errors for single elimination tournaments. Due to the fact that the double elimination format was never completed for the Auto-Scheduler, support for double elimination brackets in the Bracket Builder was not implemented.

4.3 Summary

All the test cases indicate that the modules perform as expected and to design specification. However there are some features that do contain bugs are were not implemented:

- Support for double elimination tournaments in the Auto-Scheduler is incomplete.
- Support for double elimination tournaments is not implemented in the Bracket Builder.
- The Client Software requires more error handling code to be implemented; in particular when there is no Chat Server present.
- The RCON interface has been successfully integrated into existing GotGames services.
- All modules are functional.

CHAPTER 5

CONCLUSIONS

5.1 Introduction

This chapter examines the results and outcome of this work and explores the avenues open for future work and research within this field. Security is addressed and possible fixes and exploits are examined.

5.2 Security

As with all online applications security is always a major concern in the design, development and testing phases. Notably absent from earlier sections of this text it is fitting that security be addressed within the conclusion.

It must be emphasized that the modules created throughout this project do not form an entire Tournament Management System but rather perform key applications within a whole system. There are a number of potential vulnerabilities with the Client Software:

- The windows registry could be altered to point to a fake Steam log file.
- The Steam log file could be edited to contain fake information.
- Multiple users can log onto the software using the same login details.
- Both the hashed password and the salt required to decode it are sent to the client software and could be intercepted.

There are also some features that prevent exploits in the Client and Server software:

- The use of server timestamps on every message received by the Chat server including Client connects and disconnects can be used to identify online time for individual users.
- The Server log file contains the IP addresses of each user which in turn can be used to narrow down their geographical location or compared to GotGames IP address records.

The following security enhancements could be made to the Client Software:

- The location of the steam logs could be verified by forcing users to have Steam running in order to login. This would allow a match between the Windows registry and steam process path to be determined.
- The Client could ensure that the desired Source game is open by examining the process list.
- Access could be limited to deny simultaneous access from the same user.

- Steam IDs could be read on connection to the server through the RCON protocol and compared to those entered on GotGames.

There are a number of potential exploits within the client system, while the other modules are relatively strong from a security viewpoint. Further discussion regarding possible improvements and additional source code are covered in Section 4.4 Further Work.

5.3 Conclusions

It is possible to automate many of the tedious tasks associated with tournament management. The ability to automatically schedule a tournament and supply anyone a detailed tournament bracket is a huge step forward in the automation process. Although not fully functional or a complete system the modules designed do perform their intended tasks.

Although support for double elimination brackets is lacking due to complexity and time constraints it is possible to automate a single elimination tournament using the tools created. Being able to effectively and securely authenticate a player's identity in an online environment remains an elusive task. The Client software needs improvement before the authentication methods could be considered comprehensive and secure enough to guarantee a player identity.

Design and development of online tournament management systems is still application dependant although through the use of tools explored in this project a more universal approach is possible. Extension of the PHP scripts to include a universal server booking system was omitted due to the fact that GotGames implemented a similar application during the writing of this document.

5.4 Further Work

There are a number of areas where improvement could be made to the existing modules. Implementation of a double elimination Scheduler and Bracket Builder is an obvious omission from this work.

Extensive improvement could be made to the Client and Server Applications with the potential to add anti-cheat functionality to the Client and Statistics could be stored for each individual Steam ID. The user chat functions could be improved to become room based with each match containing their own players.

A protocol needs to be developed between the Client and Server to allow the dynamic sharing of data. This would be extremely useful for retrieving scheduling information and would allow the server to request information from the Client. This two way protocol could allow polling of clients by the server and form the basis of an anti-cheat system.

Included in **APENDIX** is some additional Client source code that was developed for this project. The code allows for the Client to establish if a process is running on the host PC and to gather a list of running processes. This code is potentially useful in the development of an anti-cheat Client.

Finally this project is sponsored by a corporate company and it would be useful to explore the option of adding advertising to the Client Software. The client must be run by every player and it is therefore an ideal medium for advertising to a target demographic.

REFERENCES

McLaughlin, 2001, *JAVA and XML*, O'Riley, Cambridge.

Farrell, 2003, *JAVA Programming*, Thomson/Course Technology, Australia.

Galbraith, B & Almaer, D, 2006, *Pragmatic Ajax a Web 2.0 primer*, Pragmatic Bookshelf. NC

McKinnon, L, 2003, *XML*, Thomson Course Technology, Boston.

Durkin, K, 1999, *Computer games and Australians today*, NSW Government, Sydney.

Smed, J, 2006, *Algorithms and networking for computer games*, Wiley, England.

Source RCON Protocol, accessed 19 Oct 2009, < http://developer.valvesoftware.com>.

Jayson vs XML: The Debate, accessed 15 Aug 2009, < http://ajaxian.com/archives/json-vs-xml-the-debate>.

Debate: JSON vs. XML as a data interchange format, accessed 23 Dec 2009, <http://www.infoq.com/news/2006/12/json-vs-xml-debate>.

Paid to Play, accessed 19 Nov 2009, <http://au.gamespot.com/features/6195362/index.html>

JAVA Socket Programming in Client/Server Applications, accessed Jul 23 2009, < http://www.developer.com/java/article.php/3840466/article.htm>

Source Engine, accessed Sep 11 2009 < http://source.valvesoftware.com/>

Loy, M, 2003, *Java Swing*, O'Riely, California

CEVO Staff Instructions, accessed Dec 17 2008, <www.cevo.com>

APPENDICES

APPENDIX A

Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

FOR:	**Benjamin Brett Thomas**
TOPIC:	Development of a tournament management system
SUPERVISOR:	Alexander Kist
PROJECT AIM:	To develop a tournament management a system for online games based on Valve's source engine.
SPONSORSHIP:	GotGames
PROGRAMME:	**Issue A. 24th April 2009**

1. Research existing systems for comprehensive tournament management.

2. Research and select a secure and reliable protocol for data encapsulation and inter site communication.

3. Investigate and develop secure communication for client and server interaction to verify user identity and share match scheduling information.

4. Develop a client program that runs on the player's computer. This program communicates with the management system to verify player identity for match play.

5. Develop a server management module, capable of managing third party game servers. The module should include a booking system, be able to handle SourceTV requests; and support automated bracket building, match scheduling and finals procedures.

As time permits:

6. Extend the client program to allow users to communicate through text based chat.

7. Investigate and implement methods for recording statistics from competition matches.

8. Development a subsystem that allows players to organize teams, schedule matches and maintain personal details including game IDs.

9. Investigate basic anti-cheat measures that can be added to the client.

AGREED Ben Thomas (student) _____ (supervisor)
Date: 12/ 4 /2009 Date: 28/04 /2009

Examiner/Co-Examiner: 24/4/09 28/04/09

APPENDIX B

RCON SERVER COMMANDS

RCON Commands		
Task	**Syntax**	**Description**
RCON Login	rcon_password yourpassword	Login to rcon with your rcon password to gain access to all the rcon commands.
Change Level	changelevel map_name	This command allows you to change the map.
Users list	status	Lists the current players names with PlayerID, Name, Steamid and ip address.
Kick Player Name	kick name	If you need to kick a player replace name with the players name.
Kick Player ID	kick STEAM_0:0:123456	If you need to kick a player replace number with the players number.
Ban Player ID	banid time STEAM_0:0:123456>	If you need to ban a player replace time with the amount of time you would like to ban them. Enter 0 for a perma-ban.
Server Password	sv_password password	To password protect the server replace password with the games password.
Restart Game	mp_restartgame time	Resart the game, both teams scores are reset as is money. replace time with a value in seconds up to 10.
Frienly Fire	mp_friendlyfire number	Replace number with 1 to turn friendly fire on or 0 to turn it off.
Chase Cam	mp_forcechasecam number	Replace number with 1 to force chase cam mode or 0 to disable it.
Round Limit	mp_roundlimit number	Replace number with the amount of rounds before the level changes.
Round Time	mp_roundtime number	Replace number with a time in minutes - normally 5 (3-15 allowed).
Time Limit	mp_timelimit number	Replace number with the time in minutes for each map - 0 = no time limit.
Say something though server	say text	Sends a message from the server admin to the screen which all players can read. replace text with the message.
Color Message	cm_say @@r text	Sends a color message to the centre of the screen. replace r which is red with g for green y for yellow etc. replace text with the message.
Restart Server	quit	Restarts your server
Staring Money	mp_startmoney x	This integer value controls the amount that players start with at the beginning of a new map or when they join a server. 800 is the default and the minimum value, while 16000 is the maximum.
Deatailed Log	mp_logdetail x	Use this bitwise svar to control the depth of your logs. Be warned, these options can eat away hard drive space and CPU cycles. Use 0, the default, to log no attacks, 1 to log enemy attacks, 2 to log teammate attacks, and 3 to log both.
Player ID	mp_playerid x	This variable controls what players see when they put their crosshair over an enemy, a hostage, or a teammate. At 0 (the default), the player see's all popups with team colors. At 1, the player sees their teammates and hostages, with team colors. At 2, players see no popups.
Fading	mp_fadetoblack x	This is also to combat ghosting, but is more harsh. 0 will have no affect on the game, while 1 will disable chasecam and any sort of death cam at all. The client's screen fades to black instantly after death. One boon of this setting is that corpses stay around for the entire round. Useful for clan matches.
Buying Time	mp_buytime x	A floating point (decimal) value to determine the buy time in minutes. E.g., '1.25' equals one minute and fifteen seconds of buy time (this is not the pause, but the time where it is legal to purchase stuff). The minimum value is 0.25 and there is no maximum.
Win Limit	mp_winlimit x	This, of course, is an integer (whole number). When a team reaches this amount of wins (before the timelimit or the roundlimit is reached), it has won the map. 0, or disabled, is the default.
Spectators	allow_spectators x	Setting to 1 will allow spectators, while 0 will disallow them. These are non-playing spectators, not dead people.

Team Killer Banning	Team Killer Banning	This variable is defaulted to 1. Toggles automatic team-killer banning and kicking of idle clients - Setting of 1 = 'on', and 0 = 'off'.
Team Balancing	mp_autoteambalance x	This variable is defaulted to 1. Toggles the forcing of clients to join teams to make it balanced - Setting of 1 = 'on', and 0 = 'off'.
Bomb Timer	mp_c4timer x	This variable is defaulted to 45. Sets the amount of time in between C4 placement and its explosion - ranges between 15 and 90 seconds.
Use Flashlights	mp_flashlight x	This variable is defaulted to 1. Toggles the use of flashlights by clients - Setting of 1 = 'on', and 0 = 'off'.
Hear Footsteps	mp_footsteps x	This variable is defaulted to 1. Toggles footstep sounds - Setting of 1 = 'on', and 0 = 'off'.
Freeze Players For Buying	mp_freezetime x	This variable is defaulted to 6. Sets the amount of "freeze" time at the beginning of each round to buy weapons and equipment - Use a setting of '0' to disable.
Hostage Kill Kicking	mp_hostagepenalty x	This variable is defaulted to 0, or disabled. Sets the number of hostages a player can kill before they are booted from the server.
Number of more players on team can have over another	mp_limitteams x	This variable is defaulted to 2. Sets the maximum number of players that one team can have more than the other team - Use a setting of '0' to completely disable the team limiting.
Log Chat Messages	mp_logmessages x	This variable is defaulted to 1. Toggles logging of chat messages in the log files - Setting of 1 = 'on', and 0 = 'off'.
Map Vote Ration	Map Vote Ration	This variable is defaulted to 0.6. Set the ratio of players required to vote on the same map before a map will change. The default is 0.6 , which means that 60% of the players on a server must vote on the same map in order for the server to change to that map. The range is 0.0-1.0.
Kick Vote Ration	mp_kickpercent x	This variable is defaulted to 0.66. Set the ratio of players on someone's team required to vote to kick the "someone." The default is 0.66, which means that 66% (2/3rds) of the players on his team must vote him off in order for him to be kicked. The range is 0.0-1.0.
Team Kill Punishing	mp_tkpunish x	This variable is defaulted to 1. Toggles the forcing of a player to sit out the next round if he has just killed a teammate - Setting of 1 = 'on', and 0 = 'off'.

APPENDIX C

Bracket Builder Source Code: bracket.php

```
<?
// PHP Class bracket
// Bracket builder for GotGames
// Author Benjamin Thomas
// October 2009
class bracket {

	//Bring In Database Details From Include.PHP
	var $username = DB_USER;
	var $password = DB_PASS;
	var $database = DB_NAME;

	//Init Class Variables
	var $tournament_name;
	var $tournament_size;
	var $tournament_format;
	var $tournament_id;
	var $result_servers;
	var $result_stvs;
	var $result_matches;
	var $result_teams;



	//**************************************************************************************
*******************************
	// schedule bracket
	// Input - The tournament ID used to identify which tournament we are scheduling for
	// Operation - This constructor reads the required information from the databas and assigns the following
class values:
	// tournament_name;
	// tournament_size;
	// tournament_format;
	// tournament_id;
	// result_servers;
	// result_stvs;
	// result_matches;
	// result_teams;
	//**************************************************************************************
*******************************
	function bracket($tourn_id) {

		//Save the Tournament ID
		$this->tournament_id = $tourn_id;

		//Collect All Required Information To Draw a Bracket
		mysql_connect(localhost,$this->username,$this->password);
		@mysql_select_db($this->database) or die( "Unable to select database");

		$query  = "SELECT * FROM tms_tournament WHERE id = ".$tourn_id;
		$result_tournament = mysql_query($query) or die('Error, query failed');
		$row = mysql_fetch_assoc($result_tournament);
		$this->tournament_name = $row['name'];
		$this->tournament_size = $row['size'];
		$this->tournament_format = $row['format'];
		mysql_free_result($result_tournament);
```

```
		//read and save information from database
		$query  = "SELECT * FROM tms_servers WHERE id_tournament = ".$tourn_id;
		$this->result_servers = mysql_query($query) or die('Error, query failed');

		$query  = "SELECT * FROM tms_stvs WHERE id_tournament = ".$tourn_id;
		$this->result_stvs = mysql_query($query) or die('Error, query failed');

		$query  = "SELECT * FROM tms_matches WHERE id_tournament = ".$tourn_id;
		$this->result_matches = mysql_query($query) or die('Error, query failed');

		$query  = "SELECT * FROM tms_teams_".$tourn_id." WHERE id_tournament = ".$tourn_id;
		$this->result_teams = mysql_query($query) or die('Error, query failed');

	}

	//*****************************************************************************************
*****************************
	// Function draw
	// Operation - Called to display a single elimination bracket
	//*****************************************************************************************
*****************************
	function draw() {
		$total_rounds = log($this->tournament_size,2)+1; //total rounds
		$row = array();
		$col = array($row); //create a data type to hold our bracket information
		//Generate a datastructure to hold all the information required to layout the bracket
		for($i = 1; $i <= $total_rounds; $i++) {
			$round_matches = $this->getmatches($i);
			$matches = pow(2,$total_rounds-$i); //calc how many matches for this round
			$interval = pow(2,$i); // calc the interval for layout spacing
			$offset = pow(2,$i-1); // each round is offset by a differnt amount to form the bracket
pyramid
				for($c = 1; $c <= ($this->tournament_size*2)+1; $c++) {
					if ($c < $offset) { // blank space
						$col[$i][$c] = 0;
					} elseif ($c > (($this->tournament_size*2)+1)-$offset) { // blank space
						$col[$i][$c] = 0;
					} elseif ($c==$offset) {
						if ($i==$total_rounds) { //No match but tournament winner
						$col[$i][$c] = "Champion";
						}else { //print a team here
						$tmp_array=array_shift($round_matches);
						$col[$i][$c] = $tmp_array['print_team'];
						}
					} elseif ((($c-$offset) % $interval) == 0) { //print a team here
						$tmp_array=array_shift($round_matches);
						$col[$i][$c] = $tmp_array['print_team'];

					} elseif ($c==($offset*2)) { //print match here
						$col[$i][$c] = "Match".$tmp_array['id'];
					} elseif ((($c-$offset*2) % ($interval*2)) == 0) { //print match here
						$col[$i][$c] = "Match".$tmp_array['id'];
					} else {
						$col[$i][$c] = 0; // blank space
					}
				}
```

```
                                print("<br>");
                        }
                        // layout the bracket using html tables and the data struct created above: col
                        print("<table width='100%' border='5'>");
                        print("<tr>");
                        for($i = 1; $i <= $total_rounds-1; $i++) {
                                print("<th>Round ".(string)$i."</th>");
                        }
                        print("</tr>");
                        for($c=1;$c<=($this->tournament_size*2);$c++) {
                                print("<tr>");
                                for($i = 1; $i <= $total_rounds+1; $i++) {
                                        if (strcmp(substr($col[$i][$c],0,5),"Match")==0) {
                                                $tmp_array = $this->getmatch(substr($col[$i][$c],5));
                                                $datetime = new DateTime($tmp_array['timestamp']);
                                                print("<td align='center' bgcolor='#FFE4E1'><table><tr><td
align='center'>".$datetime->format("D, jS F Y   gA")."</td></tr><tr><td align='center'>stv - ".$this-
>getstvdetails($tmp_array['id_stv'])."</td></tr></table></td>");
                                        } elseif (strcmp(substr($col[$i][$c],0,5),"Teams")==0) {
                                                print("<td align='center' bgcolor='#dddddd'>".$this-
>getteamname(substr($col[$i][$c],5))."</td>");
                                        } elseif ($col[$i][$c]) {
                                                print("<td align='center' bgcolor='#dddddd'>".$col[$i][$c]."</td>");
                                        } else {
                                                print("<td height='40'></td>");
                                        }
                                }
                                print("</tr>");
                        }
                        print("</table>");
                }


                //*************************************************************************************
*****************************
                // Function getmatch
                // Input - match_id
                // Operation - Retrives all the information saved for the parsed match_id
                //*************************************************************************************
*****************************
                function getmatch($match_id) {
                        $query  = "SELECT * FROM tms_matches WHERE id = ".(string)$match_id;
                        $result = mysql_query($query);
                        $row = mysql_fetch_assoc($result);
                        mysql_free_result($result);
                        return $row;
                }

                //*************************************************************************************
*****************************
                // Function getteamname
                // Input - team_id
                // Operation - Returns the name of the team with the parsed team ID
                //*************************************************************************************
*****************************
                function getteamname($team_id) {
```

```
		$query  = "SELECT name FROM tms_teams_".(string)$this->tournament_id." WHERE id =
".(string)$team_id;
		$result = mysql_query($query) or die('Error, query failed');
		return mysql_result($result, 0);
	}

	//**************************************************************************************
*******************************
	// Function getstvdetails
	// Input - stv_id
	// Operation - Returns the IP Address of the parsed stv id
	//**************************************************************************************
*******************************
	function getstvdetails($stv_id) {
				$query  = "SELECT address FROM tms_stvs WHERE id = ".(string)$stv_id;
				$result = mysql_query($query);
				return mysql_result($result, 0);
	}

	//**************************************************************************************
*******************************
	// Function getmatches
	// Input round_no
	// Operation - Returns an array of all the matches for the parse round. The array is sorted by the position
field.
	//**************************************************************************************
*******************************
	function getmatches($round_no) {
		$matches_avail = array();
		$tmp_row = array();
		$query  = "SELECT * FROM tms_matches WHERE id_tournament = ".$this->tournament_id."
AND round = ".(string)$round_no." ORDER BY position";
		$result = mysql_query($query);
		while($row = mysql_fetch_assoc($result))
		{
			$tmp_row = $row;
			if ($row['status'] == 1) {
				$row = array_merge($row,array("print_team"=>"Winner of
".$row['id_match_parent_a']));
			} else {
				$row =
array_merge($row,array("print_team"=>"Teams".$row['id_team_a']));//team A
			}
			$matches_avail[] = $row;
			$row = $tmp_row;
			if ($row['status'] == 1) {
				$row = array_merge($row,array("print_team"=>"Winner of
".$row['id_match_parent_b']));
			} else {
				$row =
array_merge($row,array("print_team"=>"Teams".$row['id_team_b']));//team B
			}
			$matches_avail[] = $row;
			unset($tmp_row);
		}
		return $matches_avail;
```

```
	}

	//*************************************************************************************
*****************************
	// destructor
	// Operation - Frees memory held by SQL result sets
	//*************************************************************************************
*****************************
	function __destruct() {
		mysql_free_result($this->result_servers);
		mysql_free_result($this->result_stvs);
		mysql_free_result($this->result_matches);
		mysql_free_result($this->result_teams);
		mysql_close();
	}

}
?>
```

APPENDIX D

Auto-Scheduler Source Code: schedule.php

```
<?
// PHP Class schedule
// Tournament auto scheduler for GotGames
// Author Benjamin Thomas
// August 2009

class schedule {

        //Bring In Database Details From Include.PHP
        var $username = DB_USER;
        var $password = DB_PASS;
        var $database = DB_NAME;

        //Init Class Variables
        var $tournament_name;
        var $tournament_size;
        var $tournament_format;
        var $tournament_id;
        var $tournament_start;
        var $tournament_seed;
        var $tournament_freq;
        var $server_count;
        var $stv_count;
        var $week_dates;
        var $frequency;
        var $nights_per_interval;
        var $result_servers;
        var $result_stvs;
        var $result_matches;
        var $result_teams;

        //*********************************************************************************
*******************************
        // schedule constructor
        // Input - The tournament ID used to identify which tournament we are scheduling for
        //Operation - This constructor reads the required information from the databas and assigns the following
class values:
```

```
        // tournament_name
        // tournament_size
        // tournament_format
        // tournament_start
        // tournament_seed
        // tournament_freq
        // result_servers
        // server_count
        // result_stvs
        // stv_count
        // result_matches
        // result_teams
        //*************************************************************************************
*****************************
        function schedule($tourn_id) {
                //Save the Tournament ID
                $this->tournament_id = $tourn_id;
                $this->week_dates = $start_dates;
                $this->frequency = $interval;
                $this->nights_per_interval = $games;
                //Collect All Required Information
                //We are reading all the information from the database
                mysql_connect(localhost,$this->username,$this->password);
                @mysql_select_db($this->database) or die( "Unable to select database");
                $query  = "SELECT * FROM tms_tournament WHERE id = ".$tourn_id;
                $result_tournament = mysql_query($query) or die('Error, query failed');
                $row = mysql_fetch_assoc($result_tournament);
                $this->tournament_name = $row['name'];
                $this->tournament_size = $row['size'];
                $this->tournament_format = $row['format'];
                $this->tournament_start = $row['starttime'];
                $this->tournament_seed = $row['seeded'];
                $this->tournament_freq = $row['frequency'];
                mysql_free_result($result_tournament);
                $query  = "SELECT * FROM tms_servers WHERE id_tournament = ".$tourn_id;
                $this->result_servers = mysql_query($query) or die('Error, query failed');
                $this->server_count = mysql_num_rows($this->result_servers);
```

```
		$query  = "SELECT * FROM tms_stvs WHERE id_tournament = ".$tourn_id;
		$this->result_stvs = mysql_query($query) or die('Error, query failed');
		$this->stv_count = mysql_num_rows($this->result_stvs);
		$query  = "SELECT * FROM tms_matches WHERE id_tournament = ".$tourn_id;
		$this->result_matches = mysql_query($query) or die('Error, query failed');
		$query  = "SELECT * FROM tms_teams_".$tourn_id." WHERE id_tournament = ".$tourn_id;
		$this->result_teams = mysql_query($query) or die('Error, query failed');
	}

	//*************************************************************************************
*******************************
	// Function automate
	// Operation - Used to call the functions responsible for sutomating sheduling
	//*************************************************************************************
*******************************
	function automate() {
		$team_count = mysql_num_rows($this->result_teams) or die("No Teams Added For
Tournamnet");
		if ($team_count != $this->tournament_size) {
			die("You Must Have The Correct Number of Teams Saved!");
		}
		switch ($this->tournament_format) {
		   case "single":
		      $this->single();
		      break;
		   case "double":
		      echo "double";
		      break;
		}


	}

	//*************************************************************************************
*******************************
	// Function double
	// Operation - Called from automate and is responsible for autoscheduling a double elimination bracket.
```

```
        // NB - Incomplete!
        //*********************************************************************************************
*****************************
        function double() {
                echo "Automated Double";
                echo "<br> Stvs:";
                echo $this->stv_count;
                echo "<br> Servers: ";
                echo $this->server_count;
                $matches = $this->tournament_size / 2;
                $matches_min = ceil($matches / $this->server_count);
                echo "<br> Min Time: ";
                echo $matches_min;
                echo "<br> Start Time: <br>";
                //echo $this->tournament_start;
                $datetime = new DateTime($this->tournament_start);
                $round_time = new DateTime($datetime->format(DATE_ATOM));
                //echo $datetime->format(DATE_ATOM);
                switch ($this->tournament_freq) {
                                        case "Day":
                                           $time_gap="+1 day";
                                           break;
                                        case "Week":
                                           $time_gap="+1 week";
                                           break;
                }
                $available_stvs = $this->getstvs();
                $available_servers = $this->getservers();
                if (sizeof($available_servers) == 0 ) {
                        die ("No Servers For Tournament");
                }
                $total_rounds = log($this->tournament_size,2);
                $seeded_teams = $this->calculateseeds();
                //we only know the teams for round one then there are alot of nulls in our database! But we have
enough to manage servers!
```

```
	}

	//*************************************************************************************
*****************************
	// Function single
	// Operation - Called from automate and is responsible for autoscheduling a single elimination tournament.
	//*************************************************************************************
*****************************
	function single() {
		echo "Automated Single";
		echo "<br> Stvs:";
		echo $this->stv_count;
		echo "<br> Servers: ";
		echo $this->server_count;
		// matches for 1st round will be
		$matches = $this->tournament_size / 2;
		// calculate how much time we will need given 1 hr for each match
		$matches_min = ceil($matches / $this->server_count);
		echo "<br> Min Time: ";
		echo $matches_min;
		echo "<br> Start Time: <br>";
		//echo $this->tournament_start;
		$datetime = new DateTime($this->tournament_start);
		$round_time = new DateTime($datetime->format(DATE_ATOM));
		//echo $datetime->format(DATE_ATOM);
		// calculate the time difference for each round using the DateTime object
		switch ($this->tournament_freq) {
				case "Day":
				   $time_gap="+1 day";
				   break;
				case "Week":
				   $time_gap="+1 week";
				   break;
		}
		// get available STVs
		$available_stvs = $this->getstvs();
		// get available dervers
```

```
$available_servers = $this->getservers();
if (sizeof($available_servers) == 0 ) {
        die ("No Servers For Tournament");
}
// calculate the number of rounds required
$total_rounds = log($this->tournament_size,2);
// calcualte seeds
$seeded_teams = $this->calculateseeds();
//we only know the teams for round one then there are alot of nulls in our database! But we have enough to manage servers!
for($i = 1; $i <= $total_rounds; $i++) {
        //lets use our seeded teams array to make this easy! well easier thanks to PHP allowing us to pop and shift
        $matches = pow(2,$total_rounds-$i);
        //$tmp_server_count=$this->server_count;
        $pos_high = 1;
        $pos_low = $matches;
        if ($i==1) { //Round 1
                $counter = 0;
                while (sizeof($seeded_teams) > 0) {
                        $assigned_server = array_pop($available_servers);
                        $assigned_stv = array_pop($available_stvs);
                        if ($assigned_server==NULL) { //grab a server and stv
                                $available_servers = $this->getservers();
                                $available_stvs = $this->getstvs();
                                $round_time->modify("+1 hour");
                                $assigned_server = array_pop($available_servers);
                                $assigned_stv = array_pop($available_stvs);
                        }
                        $teama=array_shift($seeded_teams);
                        $teamb=array_pop($seeded_teams);
                        if (($counter % 2) == 0) {
                                //high
                                $position = $pos_high;
                                $pos_high++;
                        } else {
                                //low
```

```
                                        $position = $pos_low;
                                        $pos_low--;
                                }
                                $counter++;
                                echo "Round:".(string)$i." Match:".(string)$teama."V".(string)$teamb."
Time:".$round_time->format(DATE_ATOM)." Tourn: ".$this->tournament_id." Pass:".(string)$this-
>generatepass()." Server: ".(string)$assigned_server." STV: ".(string)$assigned_stv." Pos: ".$position."<br>";
                                //save match
                                $match_info =
array("round"=>(string)$i,"id_teama"=>(string)$teama,"id_teamb"=>(string)$teamb,"timestamp"=>(string)$round_t
ime->format(DATE_ATOM),"id_tournament"=>(string)$this->tournament_id,"server_pw"=>(string)$this-
>generatepass(),"id_server"=>(string)$assigned_server,"id_stv"=>(string)$assigned_stv,"position"=>(string)$positi
on);
                                //print_r($match_info);
                                $this->savematch($match_info,TRUE);
                        }
                } else {
                        $prev_rnd_matches = $this->getroundmatches($i-1);
                        print_r($prev_rnd_matches);
                        $counter=0;
                        for($c=0;$c<$matches;$c++) { //get a server and stv
                                $assigned_server = array_pop($available_servers);
                                $assigned_stv = array_pop($available_stvs);
                                if ($assigned_server==NULL) {
                                        $available_servers = $this->getservers();
                                        $available_stvs = $this->getstvs();
                                        $round_time->modify("+1 hour");
                                        $assigned_server = array_pop($available_servers);
                                        $assigned_stv = array_pop($available_stvs);
                                }
                                $counter++;
                                $parent_match_a = array_shift($prev_rnd_matches);
                                $parent_match_b = array_shift($prev_rnd_matches);

                                //save match
                                echo "Round:".(string)$i." Match:".(string)$c." Time:".$round_time-
>format(DATE_ATOM)." Tourn: ".$this->tournament_id." Pass:".(string)$this->generatepass()." Server:
```

```
".(string)$assigned_server." STV: ".(string)$assigned_stv." Pos: ".(string)$counter."ParentA:
".(string)$parent_match_a." ParentB: ".(string)$parent_match_b."<br>";
                                        $match_info =
array("round"=>(string)$i,"timestamp"=>(string)$round_time-
>format(DATE_ATOM),"id_tournament"=>(string)$this->tournament_id,"server_pw"=>(string)$this-
>generatepass(),"id_server"=>(string)$assigned_server,"id_stv"=>(string)$assigned_stv,"position"=>(string)$count
er,"id_match_parent_a"=>(string)$parent_match_a,"id_match_parent_b"=>(string)$parent_match_b);
                                        $this->savematch($match_info,FALSE);
                                }
                        }
                $datetime->modify((string)$time_gap);
                $round_time = new DateTime($datetime->format(DATE_ATOM));
                $available_stvs = $this->getstvs();
                $available_servers = $this->getservers();
                }
        }

        //**********************************************************************************************
******************************
        // Function savematch
        // Inputs:
        // match_detials - Multi-demensional array holding all the match information to be saved
        // teams_known - Set to true is we know the names of the participating teams otherwise set to false.
        // Operation - This functions writes the match information to the database
        //**********************************************************************************************
******************************
        function savematch($match_details, $teams_known) {
                if ($teams_known) { // we know the teams
                        if ($match_details['id_stv'] == NULL) { //no stv for match
                                $query = "INSERT INTO
tms_matches(round,id_team_a,id_team_b,timestamp,id_tournament,server_pw,id_server,position)
Values(".(string)$match_details['round'].",".(string)$match_details['id_teama'].",".(string)$match_details['id_teamb']
.",'".(string)$match_details['timestamp']."',".(string)$match_details['id_tournament'].",'".(string)$match_details['serv
er_pw']."',".(string)$match_details['id_server'].",".(string)$match_details['position'].")";
                        } else {
                                $query = "INSERT INTO
tms_matches(round,id_team_a,id_team_b,timestamp,id_tournament,server_pw,id_server,id_stv,position)
```

```
Values(".(string)$match_details['round'].",".(string)$match_details['id_teama'].",".(string)$match_details['id_teamb'].",'".(string)$match_details['timestamp']."',".(string)$match_details['id_tournament'].",'".(string)$match_details['server_pw']."',".(string)$match_details['id_server'].",".(string)$match_details['id_stv'].",".(string)$match_details['position'].")";
                                }
                                echo $query;
                                echo "<br>";
                                mysql_query($query) or die('Error, query failed');
                        } else {
                                if ($match_details['id_stv'] == NULL) { //no stv for match
                                        $query = "INSERT INTO
tms_matches(round,timestamp,id_tournament,server_pw,id_server,position,id_match_parent_a,id_match_parent_b,status)
Values(".(string)$match_details['round'].",'".(string)$match_details['timestamp']."',".(string)$match_details['id_tournament'].",'".(string)$match_details['server_pw']."',".(string)$match_details['id_server'].",".(string)$match_details['position'].",".(string)$match_details['id_match_parent_a'].",".(string)$match_details['id_match_parent_b'].",1)";
                                } else {
                                        $query = "INSERT INTO
tms_matches(round,timestamp,id_tournament,server_pw,id_server,id_stv,position,id_match_parent_a,id_match_parent_b,status)
Values(".(string)$match_details['round'].",'".(string)$match_details['timestamp']."',".(string)$match_details['id_tournament'].",'".(string)$match_details['server_pw']."',".(string)$match_details['id_server'].",".(string)$match_details['id_stv'].",".(string)$match_details['position'].",".(string)$match_details['id_match_parent_a'].",".(string)$match_details['id_match_parent_b'].",1)";
                                }
                                echo $query;
                                echo "<br>";
                                mysql_query($query) or die('Error, query failed');
                        }
                }

                //*************************************************************************************
*******************************
                // Function getservers
                // Operation - called to get an array of server ids
                //*************************************************************************************
*******************************
```

```
	function getservers() {
		$servers_avail = array();
		while($row = mysql_fetch_assoc($this->result_servers))
		{
			$servers_avail[] = $row['id'];
		}
		mysql_data_seek($this->result_servers,0);
		return $servers_avail;
	}

	//*************************************************************************************
*******************************
	// Function getstvs
	// Operation - called to get an array of stv ids
	//*************************************************************************************
*******************************
	function getstvs() {
		$stvs_avail = array();
		while($row = mysql_fetch_assoc($this->result_stvs))
		{
			$stvs_avail[] = $row['id'];
		}
		mysql_data_seek($this->result_stvs,0);
		return $stvs_avail;
	}

	//*************************************************************************************
*******************************
	// Function getroundmatches
	// Input - rnd_no - the number of the round.
	// Operation - called to get the id of round matches
	//*************************************************************************************
*******************************
	function getroundmatches($rnd_no) {
		$rnd_matches = array();
		$query  = "SELECT id FROM tms_matches WHERE round = ".(string)$rnd_no." AND
id_tournament = ".(string)$this->tournament_id." ORDER BY position ASC";
```

```
		$result = mysql_query($query) or die('Error, query failed');
		while($row = mysql_fetch_assoc($result))
		{
			$rnd_matches[] = $row['id'];
		}
		return $rnd_matches;

	}

	//*************************************************************************************
*******************************
	// Function caculateseeds
	// Operation - called to get an array of seeded teams or if unseeded randomly shuffles the returned team
array
	//*************************************************************************************
*******************************
	function calculateseeds () {
		$seeded_teams = array();
		if ($this->tournament_seed) {
				while($row = mysql_fetch_assoc($this->result_teams))
				{
					$seeded_teams[((int)$row['seed'])-1] = $row['id'];
				}
				//seeded
			} else {
				$c=1;
				while($row = mysql_fetch_assoc($this->result_teams))
				{
					$seeded_teams[$c] = $row['id'];
					$c++;
				}
				shuffle($seeded_teams);
				//unseeded
			}
			mysql_data_seek($this->result_teams,0);
			return $seeded_teams;
		}
```

```
	//*************************************************************************************
*******************************
	// Function generatepass
	// Operation - called to generate a random password used to set server passwords
	//*************************************************************************************
*******************************
	function generatepass() {
		$chars = "abcdefghijkmnopqrstuvwxyz023456789";
		srand((double)microtime()*1000000);
		$i = 0;
		$pass = '' ;
		while ($i <= 7) {
			$num = rand() % 33;
			$tmp = substr($chars, $num, 1);
			$pass = $pass . $tmp;		$i++;
		}
		return $pass;
	}

	//*************************************************************************************
*******************************
	// destructor
	// Operation - Frees memory held by SQL result sets
	//*************************************************************************************
*******************************
	function __destruct() {
			mysql_free_result($this->result_servers);
			mysql_free_result($this->result_stvs);
			mysql_free_result($this->result_matches);
			mysql_free_result($this->result_teams);
			mysql_close();
	}
}
?>
```

APPENDIX E

Temporary Interface Scripts for Data Manipulation

Addserver.php

Create.php

Teamselect.php

Addserver.php

```
// PHP script to add a server to the test data
// Author: Benjamin Thomas
<?
	if (isset($_POST['submit'])) {
	//save a new tournament in the database
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");

	//store tournament
	$q = "SELECT id_game FROM tms_tournament WHERE id ="(string)$_POST['tournament'];
	$r=mysql_query($q) or die('Error, query failed');
	$id_game = mysql_result($r, 0);
	if ($_POST['stv'] == 0) {
		$query = "INSERT INTO tms_servers(id_game,id_tournament,address,rcon)
Values(".(string)$id_game.",".(string)$_POST['tournament'].",'".(string)$_POST['IP']."','".(string)$_POST['rcon']."')
";
	} else {
		$query = "INSERT INTO tms_stvs(id_game,id_tournament,address,rcon)
Values(".(string)$id_game.",".(string)$_POST['tournament'].",'".(string)$_POST['IP']."','".(string)$_POST['rcon']."')
";
	}
	mysql_query($query) or die('Error, query failed');
	echo "Server Added...";
	mysql_close();
	} else {
?>
<html>
 <head>
  <title>Add Server</title>
 </head>
  <body>
 <?
	//grab any database details we will need
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");

	//grab games list
	$query  = "SELECT * FROM tms_tournament";
	$tournamentlist = mysql_query($query) or die('Error, query failed');
	mysql_close();

?>
 <p>
<form action="<?=$_SERVER['PHP_SELF']?>" method="post">
Select a Tournament:
<select name="tournament">
	<?
	while($row = mysql_fetch_assoc($tournamentlist))
		{
			print("<option value=\"{$row[id]}\">{$row[name]}</option>");
```

```
                    }
            ?>
</select>
<br>
<br>
<Input type = 'Radio' name ='stv' value= '0' >Server
<Input type = 'Radio' name ='stv' value= '1' >STV
<br>
<br>
IP:
<input type="text" name="IP" size="25" maxlength="25" />
<br>
<br>
Rcon Password:
<input type="text" name="rcon" size="64" maxlength="64" />
<br>
<br>
<input type="submit" value="Add" name="submit"/>
</form>
</p>
</body>
</html>
<?
}
?>
```

Create.php

```
// PHP script to create a tournament
// Author Benjamin Thomas
 <?
	if (isset($_POST['submit'])) {
	//save a new tournament in the database
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");
	$stime=$_POST['tournamentdate']." ".$_POST['tournamenttime'].":00:00";
	if ($_POST['seed']) {
		$seeded = 1;
	} else {
		$seeded=0;
	}
	//store tournament
	$query  = "INSERT INTO tms_tournament(id_game,name,format,size,status,starttime,seeded,frequency)
Values(".(string)$_POST['game'].",'".(string)$_POST['tournamentname']."','".(string)$_POST['tournamenttype']."','".
(string)$_POST['teamnumber']."',1,'".$stime."',".$seeded.",'".(string)$_POST['freq']."')";
	mysql_query($query) or die('Error, query failed');
	echo $query;
	echo "<br> Tournament Created...";
	mysql_close();
	//$timestring =
	//$datetime = new DateTime($_POST['tournamentdate']." ".$_POST['tournamenttime'].":00:00");
	//echo $datetime->format(DATE_ATOM);

	} else {
?>
<html>
 <head>
  <title>New Tournament</title>
 </head>
  <body>

 <?
	//grab any database details we will need
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");

	//grab games list
	$query  = "SELECT * FROM gg_games";
	$gamelist = mysql_query($query) or die('Error, query failed');
	mysql_close();
?>
 <p>
<form action="<?=$_SERVER['PHP_SELF']?>" method="post">
Select a Game:
<select name="game">
	<?
	while($row = mysql_fetch_assoc($gamelist))
		{
			print("<option value=\"{$row[id]}\">{$row[name]}</option>");
```

```
                }
        ?>

</select>
<br>
<br>
How Many Teams:
<select name="teamnumber">
        <option value="4">4</option>
        <option value="8">8</option>
        <option value="16">16</option>
        <option value="32">32</option>
        <option value="64">64</option>
</select>
<br>
<br>
Tournament Description:
<input type="text" name="tournamentname" size="128" maxlength="128" />
<br>
<br>
Tournament Type:
<select name="tournamenttype">
        <option value="single">Single Elimination</option>
        <option value="double">Double Elimination</option>
        <option value="group">Group Stages</option>
        <option value="league">League Play</option>
</select>
<br>
<br>
Start Date YYYY-MM-DD
<input type="text" name="tournamentdate" size="10" maxlength="10" />
<br>
<br>
Start Time HH (24hr Time)
<input type="text" name="tournamenttime" size="2" maxlength="2" />
<br>
<br>
<Input type = 'Radio' name ='seed' value= '1' >Seeded
<br>
<br>
Matches to be played:
<select name="freq">
        <option value="Hour">Hourly</option>
        <option value="Day">Daily</option>
        <option value="Week">Weekly</option>
</select>
<br>
<br>
<input type="submit" value="Create" name="submit"/>
</form>
</p>
</body>
</html>
<?
}
?>
```

teamselect.php

```
// PHP Script to add a team to a tournament
// Author Benjamin Thomas
<?
	if (isset($_POST['submit'])) {
	//save a new tournament in the database
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");

	$query  = "SELECT name FROM gg_teams WHERE id = ".$_POST['teams'];
	echo $query;
	$result = mysql_query($query) or die('Error, query failed');


	$team_name = mysql_result($result, 0);
	$query = "INSERT INTO tms_teams_".$_POST["t_id"]."(id,seed,id_tournament,name)
Values(".$_POST['teams'].",".(string)$_POST['seed'].",".$_POST["t_id"].",'".(string)$team_name."')";
	echo $query;
	$result = mysql_query($query) or die('Error, query failed');



	mysql_close();
	header("Location: teamselect.php");
} else {
?>
<html>
 <head>
  <title>Add Server</title>
 </head>
  <body>
 <?
	//grab any database details we will need
	include 'include.php';
	mysql_connect(localhost,$username,$password);
	@mysql_select_db($database) or die( "Unable to select database");

	//grab games list
	$query  = "SELECT * FROM tms_tournament";
	$tournamentlist = mysql_query($query) or die('Error, query failed');
	mysql_close();

?>
 <p>
<form action="<?=$_SERVER['PHP_SELF']?>" method="post">
<?	if (!isset($_POST['select'])) {
?>
Select a Tournament:
<select name="tournament">
	<?
	while($row = mysql_fetch_assoc($tournamentlist))
		{
			print("<option value=\"{$row[id]}\">{$row[name]}</option>");
```

```
                }
        ?>
</select>
<br>
<br>
<input type="submit" value="Select" name="select"/>
<?
}
?>
<?
        if (isset($_POST['select'])) {
                //grab any database details we will need
                mysql_connect(localhost,$username,$password);
                @mysql_select_db($database) or die( "Unable to select database");

                $query = "SELECT id_game FROM tms_tournament WHERE id
=".(string)$_POST['tournament'];
                $result=mysql_query($query) or die('Error, query failed');
                $id_game = mysql_result($result, 0);

                //grab games list
                $query  = "SELECT * FROM gg_teams WHERE id_game = ".(string)$id_game;
                $teamlist = mysql_query($query) or die('Error, query failed');

                $query  = "SELECT * FROM tms_teams_".(string)$_POST['tournament'];
                $teams = mysql_query($query);

                if (!$teams) {
                        $query = "CREATE TABLE  `gotgames`.`tms_teams_".(string)$_POST['tournament']."`
(`id` int(11) unsigned NOT NULL,`seed` int(11),`id_tournament` int(11) NOT NULL,`name` varchar(32) NOT
NULL,PRIMARY KEY (`id`))";
                        mysql_query($query) or die('Error, query failed');

                }

                mysql_close();
?>
<br>
<?
        mysql_connect(localhost,$username,$password);
        @mysql_select_db($database) or die( "Unable to select database");

        $query = "SELECT * FROM tms_teams_".(string)$_POST['tournament'];
        $result = mysql_query($query);
        print("<table><tr><td>TEAM</td><td>SEED</td></tr>");
        while($row = mysql_fetch_assoc($result))
        {
                print("<tr><td>".(string)$row['name']."</td><td>".(string)$row['seed']."</td></tr>");
        }
        mysql_close();
        print("</table>");
?>
<br>
<select name="teams">
        <?
        while($row = mysql_fetch_assoc($teamlist))
```

```
                {
                        print("<option value=\"{$row[id]}\">{$row[name]}</option>");
                }
        ?>
</select>
 Seed:
<input type="text" name="seed" size="3" maxlength="3" value="0" />
<br>
<br>
<input type="submit" value="Add" name="submit"/>
<input type="hidden" name="t_id" value="<?echo (string)$_POST['tournament']; ?>">
</form>
</p>
</body>
</html>
<?
}
}
?>
```

APPENDIX F

RCON Interface Source Code

RconSource.java

```
//Authors: Benjamin Thomas and Rob Skillington

package rconinter;

//import expextion handlers for this package
import rconinter.exp.BadRcon;
import rconinter.exp.ResponseEmpty;

//import java libraries
import java.io.*;
import java.net.*;
import java.nio.*;

//*************************************************************************
// Class - RconSource
// Purpose - This class is designed to implement the RCON protocol
//*************************************************************************
public class RconSource {

   // define constants and declare vars
   final static int RESPONSE_TIMEOUT = 2000;
   final static int PACKETS_TIMEOUT = 300;

   static Socket rconSocket = null;
   static InputStream in = null;
   static OutputStream out = null;

   final static int EXECCOMMAND = 2;
   final static int AUTH = 3;
   final static int RESPONSE_VALUE = 0;
   final static int AUTH_RESPONSE = 2;


    //*************************************************************************
    // constructPacket
    // Purpose - Constructs an RCON Command ot Auth packer given the parsed data
    //*************************************************************************
    private static byte[] contructPacket(int id, int cmdtype, String s1) {

      ByteBuffer p = ByteBuffer.allocate(s1.length() + 16);
      p.order(ByteOrder.LITTLE_ENDIAN);
      //length
      p.putInt(s1.length() + 12);
      //id
      p.putInt(id);
      //command type
      p.putInt(cmdtype);
      //command
      p.put(s1.getBytes());
      // two null bytes at the end
      p.put((byte) 0x00);
      p.put((byte) 0x00);
      // null string2
      p.put((byte) 0x00);
      p.put((byte) 0x00);
      return p.array();
```

```
    }

    //*************************************************************************
    // send
    // Purpose - Sends the command packet to the specified server after Auth'ing
    // with the supplied RCON password.
    //*************************************************************************
    public static String send(String ipAdd, int port, String password, String command, int localPort) throws
SocketTimeoutException, BadRcon, ResponseEmpty {
        String response = "";

        try {
            //open new socket
            rconSocket = new Socket();

            InetAddress addr = InetAddress.getLocalHost();
            byte[] ipAddr = addr.getAddress();
            InetAddress inetLocal = InetAddress.getByAddress(ipAddr);
            //try connecting
            rconSocket.bind(new InetSocketAddress(inetLocal, localPort));
            rconSocket.connect(new InetSocketAddress(ipAdd, port), 1000);

            out = rconSocket.getOutputStream();
            in = rconSocket.getInputStream();

            rconSocket.setSoTimeout(RESPONSE_TIMEOUT);

            if (rcon_auth(password)) {
                // We are now authed
                ByteBuffer[] resp = sendCommand(command);
                // Close socket handlers, we don't need them more
                out.close(); in.close(); rconSocket.close();
                if (resp != null) {
                    response = assemblePackets(resp);
                    if (response.length() == 0) {
                        throw new ResponseEmpty();
                    }
                }
            }
            else {
                throw new BadRcon();
            }
        } catch (SocketTimeoutException timeout) {
            throw timeout;
        } catch (UnknownHostException e) {
            System.err.println("UnknownHostException: " + e.getCause());
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection: "+ e.getCause());
        }

        return response;
    }

    //*************************************************************************
    // sendcommand
    // Purpose - This class sends a command and recieves the response for the server
```

```
//*************************************************************************
private static ByteBuffer[] sendCommand(String command) throws SocketTimeoutException {

   byte[] request = contructPacket(2, EXECCOMMAND, command);

   ByteBuffer[] resp = new ByteBuffer[128];
   int i = 0;
   try {
      out.write(request);
      resp[i] = receivePacket();  // First and maybe the unique response packet
      try {
         // We don't know how many packets will return in response, so we'll
         // read() the socket until TimeoutException occurs.
         rconSocket.setSoTimeout(PACKETS_TIMEOUT);
         while (true) {
            resp[++i] = receivePacket();
         }
      } catch (SocketTimeoutException e) {
         // No more packets in the response, go on
         return resp;
      }

   } catch (SocketTimeoutException timeout) {
      // Timeout while connecting to the server
      throw timeout;
   } catch (Exception e2) {
      System.err.println("I/O error on socket\n");
   }
   return null;
}

//*************************************************************************
// recievePacket()
// Purpose - Unpacks the packet once it is recieved
//*************************************************************************
private static ByteBuffer receivePacket() throws Exception {

   ByteBuffer p = ByteBuffer.allocate(4120);
   p.order(ByteOrder.LITTLE_ENDIAN);

   byte[] length = new byte[4];

   if (in.read(length, 0, 4) == 4) {
      // Now we've the length of the packet, let's go read the bytes
      p.put(length);
      int i = 0;
      while (i < p.getInt(0)) {
         p.put((byte) in.read());
         i++;
      }
      return p;
   }
   else {
      return null;
   }
}
```

```
    //*************************************************************************
    // assemblePackets
    // Purpose - Grab the text from the response packets
    //*************************************************************************
    private static String assemblePackets(ByteBuffer[] packets) {
    // Return the text from all the response packets together

        String response = "";

        for (int i = 0; i < packets.length; i++) {
            if (packets[i] != null) {
                response = response.concat(new String(packets[i].array(), 12, packets[i].position()-14));
            }
        }
        return response;
    }

    //*************************************************************************
    //  rcon_auth
    // Purpose - Sends the Auth command so that following commands can be executed
    //*************************************************************************
    private static boolean rcon_auth(String rcon_password) throws SocketTimeoutException {

        byte[] authRequest = contructPacket(1337, AUTH, rcon_password);

        ByteBuffer response = ByteBuffer.allocate(64);
        try {
            out.write(authRequest);
            response = receivePacket(); // junk response packet
            response = receivePacket();

            // Lets see if the received request_id is leet enougth ;)
            if ((response.getInt(4) == 1337) && (response.getInt(8) == AUTH_RESPONSE)) {
                return true;
            }
        } catch (SocketTimeoutException timeout) {
            throw timeout;
        } catch (Exception e) {
            System.err.println("I/O error on socket\n");
        }

        return false;
    }
    //*************************************************************************
    // send
    // Purpose - same as the orgiinal send expect the origin port is not defined
    //*************************************************************************
    public static String send(String ipAdd, int port, String password, String command) throws
SocketTimeoutException, BadRcon, ResponseEmpty {
        return send(ipAdd, port, password, command, 0);
    }

}
```

APPENDIX G

Chat Server Source Code

Server.java

ServerThread.java

Server.java

```
// Parts implemented by IBM
// Modified by Benjamin Thomas
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.Calendar;
import java.text.SimpleDateFormat;

public class Server
{

 public static final String DATE_FORMAT_NOW = "yyyy-MM-dd HH:mm:ss";

 public static String now() {
   Calendar cal = Calendar.getInstance();
   SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
   return sdf.format(cal.getTime());
 }
// The ServerSocket we'll use for accepting new connections
private ServerSocket ss;
// A mapping from sockets to DataOutputStreams. This will
// help us avoid having to create a DataOutputStream each time
// we want to write to a stream.
private Hashtable outputStreams = new Hashtable();
// Constructor and while-accept loop all in one.
public Server( int port ) throws IOException {
// All we have to do is listen
listen( port );
}
private void listen( int port ) throws IOException {
// Create the ServerSocket
ss = new ServerSocket( port );
// Tell the world we're ready to go
System.out.println( "Listening on "+ss );
// Keep accepting connections forever
while (true) {
// Grab the next incoming connection
Socket s = ss.accept();
// Tell the world we've got it
System.out.println(this.now()+" Connection from "+s );
// Create a DataOutputStream for writing data to the
// other side
DataOutputStream dout = new DataOutputStream( s.getOutputStream() );
// Save this stream so we don't need to make it again
outputStreams.put( s, dout );
// Create a new thread for this connection, and then forget
// about it
new ServerThread( this, s );
}
}
// Get an enumeration of all the OutputStreams, one for each client
// connected to us
```

```
Enumeration getOutputStreams() {
return outputStreams.elements();
}
// Send a message to all clients (utility routine)
void sendToAll( String message ) {
// We synchronize on this because another thread might be
// calling removeConnection() and this would screw us up
// as we tried to walk through the list
synchronized( outputStreams ) {
// For each client ...
for (Enumeration e = getOutputStreams(); e.hasMoreElements(); ) {
   // ... get the output stream ...
DataOutputStream dout = (DataOutputStream)e.nextElement();
// ... and send the message
try {
dout.writeUTF( message );
} catch( IOException ie ) { System.out.println( ie ); }
}
}
}
// Remove a socket, and it's corresponding output stream, from our
// list. This is usually called by a connection thread that has
// discovered that the connectin to the client is dead.
void removeConnection( Socket s ) {
// Synchronize so we don't mess up sendToAll() while it walks
// down the list of all output streamsa
synchronized( outputStreams ) {
// Tell the world
System.out.println(this.now()+" Removing connection to "+s );
// Remove it from our hashtable/list
outputStreams.remove( s );
// Make sure it's closed
try {
s.close();
} catch( IOException ie ) {
System.out.println( "Error closing "+s );
ie.printStackTrace();
}
}
}
// Main routine
// Usage: java Server <port>
static public void main( String args[] ) throws Exception {
// Get the port # from the command line
int port = Integer.parseInt( args[0] );
   //int port = 10000;
// Create a Server object, which will automatically begin
// accepting connections.
new Server( port );
}
}
```

ServerThread.Java

```
// Parts implemented by IBM
// Modified by Benjamin Thomas

import java.io.*;
import java.net.*;
import java.util.Calendar;
import java.text.SimpleDateFormat;

public class ServerThread extends Thread
{

  public static final String DATE_FORMAT_NOW = "yyyy-MM-dd HH:mm:ss";

  public static String now() {
    Calendar cal = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
    return sdf.format(cal.getTime());
  }
// The Server that spawned us
private Server server;
// The Socket connected to our client
private Socket socket;
// Constructor.
public ServerThread( Server server, Socket socket ) {
// Save the parameters
this.server = server;
this.socket = socket;
// Start up the thread
start();
}
// This runs in a separate thread when start() is called in the
// constructor.
public void run() {
try {
// Create a DataInputStream for communication; the client
// is using a DataOutputStream to write to us
DataInputStream din = new DataInputStream( socket.getInputStream() );
// Over and over, forever ...
while (true) {
// ... read the next message ...
String message = din.readUTF();
// ... tell the world ...
System.out.println(this.now()+" Sending "+message );
// ... and have the server send it to all clients
server.sendToAll( message );
}
} catch( EOFException ie ) {
// This doesn't need an error message
} catch( IOException ie ) {
// This does; tell the world!
```

```
ie.printStackTrace();
} finally {
// The connection is closed for one reason or another,
// so have the server dealing with it
server.removeConnection( socket );
}
}
}
```

APPENDIX H

Client Source Code

TMSClientFinalAboutBox.java

TMSClientFinalApp.java

TMSClientFinalMain.java

TMSClientFinalView.java

TMSLocalUser.java

TMSClientFinalAboutBox.java

```
/*
 * TMSClientFinalAboutBox.java
 */

package tmsclientfinal;

import org.jdesktop.application.Action;

public class TMSClientFinalAboutBox extends javax.swing.JDialog {

    public TMSClientFinalAboutBox(java.awt.Frame parent) {
        super(parent);
        initComponents();
        getRootPane().setDefaultButton(closeButton);
    }

    @Action public void closeAboutBox() {
        dispose();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        closeButton = new javax.swing.JButton();
        javax.swing.JLabel appTitleLabel = new javax.swing.JLabel();
        javax.swing.JLabel versionLabel = new javax.swing.JLabel();
        javax.swing.JLabel appVersionLabel = new javax.swing.JLabel();
        javax.swing.JLabel vendorLabel = new javax.swing.JLabel();
        javax.swing.JLabel appVendorLabel = new javax.swing.JLabel();
        javax.swing.JLabel homepageLabel = new javax.swing.JLabel();
        javax.swing.JLabel appHomepageLabel = new javax.swing.JLabel();
        javax.swing.JLabel appDescLabel = new javax.swing.JLabel();
        javax.swing.JLabel imageLabel = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(tmsclientfinal.TMSClientFinalApp.class).getContext().getResourc
eMap(TMSClientFinalAboutBox.class);
        setTitle(resourceMap.getString("title")); // NOI18N
        setModal(true);
        setName("aboutBox"); // NOI18N
        setResizable(false);

        javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(tmsclientfinal.TMSClientFinalApp.class).getContext().getActionM
ap(TMSClientFinalAboutBox.class, this);
        closeButton.setAction(actionMap.get("closeAboutBox")); // NOI18N
        closeButton.setName("closeButton"); // NOI18N
```

```
        appTitleLabel.setFont(appTitleLabel.getFont().deriveFont(appTitleLabel.getFont().getStyle() |
java.awt.Font.BOLD, appTitleLabel.getFont().getSize()+4));
        appTitleLabel.setText(resourceMap.getString("Application.title")); // NOI18N
        appTitleLabel.setName("appTitleLabel"); // NOI18N

        versionLabel.setFont(versionLabel.getFont().deriveFont(versionLabel.getFont().getStyle() |
java.awt.Font.BOLD));
        versionLabel.setText(resourceMap.getString("versionLabel.text")); // NOI18N
        versionLabel.setName("versionLabel"); // NOI18N

        appVersionLabel.setText(resourceMap.getString("Application.version")); // NOI18N
        appVersionLabel.setName("appVersionLabel"); // NOI18N

        vendorLabel.setFont(vendorLabel.getFont().deriveFont(vendorLabel.getFont().getStyle() |
java.awt.Font.BOLD));
        vendorLabel.setText(resourceMap.getString("vendorLabel.text")); // NOI18N
        vendorLabel.setName("vendorLabel"); // NOI18N

        appVendorLabel.setText(resourceMap.getString("Application.vendor")); // NOI18N
        appVendorLabel.setName("appVendorLabel"); // NOI18N

        homepageLabel.setFont(homepageLabel.getFont().deriveFont(homepageLabel.getFont().getStyle() |
java.awt.Font.BOLD));
        homepageLabel.setText(resourceMap.getString("homepageLabel.text")); // NOI18N
        homepageLabel.setName("homepageLabel"); // NOI18N

        appHomepageLabel.setText(resourceMap.getString("Application.homepage")); // NOI18N
        appHomepageLabel.setName("appHomepageLabel"); // NOI18N

        appDescLabel.setText(resourceMap.getString("appDescLabel.text")); // NOI18N
        appDescLabel.setName("appDescLabel"); // NOI18N

        imageLabel.setIcon(resourceMap.getIcon("imageLabel.icon")); // NOI18N
        imageLabel.setName("imageLabel"); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(imageLabel)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(versionLabel)
                            .addComponent(vendorLabel)
                            .addComponent(homepageLabel))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(appVersionLabel)
                            .addComponent(appVendorLabel)
                            .addComponent(appHomepageLabel)))
                    .addComponent(appTitleLabel, javax.swing.GroupLayout.Alignment.LEADING)
```

```
                .addComponent(appDescLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 266, Short.MAX_VALUE)
                .addComponent(closeButton))
            .addContainerGap())
        );
        layout.setVerticalGroup(
           layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addComponent(imageLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
           .addGroup(layout.createSequentialGroup()
              .addContainerGap()
              .addComponent(appTitleLabel)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addComponent(appDescLabel)
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                 .addComponent(versionLabel)
                 .addComponent(appVersionLabel))
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                 .addComponent(vendorLabel)
                 .addComponent(appVendorLabel))
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
              .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                 .addComponent(homepageLabel)
                 .addComponent(appHomepageLabel))
              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 19, Short.MAX_VALUE)
              .addComponent(closeButton)
              .addContainerGap())
        );

        pack();
    }// </editor-fold>

    // Variables declaration - do not modify
    private javax.swing.JButton closeButton;
    // End of variables declaration
```

TMSClientFinalApp.java

```
/*
 * TMSClientFinalApp.java
 */

package tmsclientfinal;

import org.jdesktop.application.Application;
import org.jdesktop.application.SingleFrameApplication;

/**
 * The main class of the application.
 */
public class TMSClientFinalApp extends SingleFrameApplication {

    /**
     * At startup create and show the main frame of the application.
     */
    @Override protected void startup() {
        show(new TMSClientFinalView(this));
    }

    /**
     * This method is to initialize the specified window by injecting resources.
     * Windows shown in our application come fully initialized from the GUI
     * builder, so this additional configuration is not needed.
     */
    @Override protected void configureWindow(java.awt.Window root) {
    }

    /**
     * A convenient static getter for the application instance.
     * @return the instance of TMSClientFinalApp
     */
    public static TMSClientFinalApp getApplication() {
        return Application.getInstance(TMSClientFinalApp.class);
    }

    /**
     * Main method launching the application.
     */
    public static void main(String[] args) {
        launch(TMSClientFinalApp.class, args);
    }
}
```

TMSClientFinalMain.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * TMSClientFinalMain.java
 *
 * Created on 21/10/2009, 5:34:41 AM
 * Benjamin Thomas
 */

package tmsclientfinal;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import TMSLocalUser.*;
/**
 *
 * @author Racs
 */
public class TMSClientFinalMain extends javax.swing.JFrame implements Runnable {

    public String Un = null;
    private Socket socket;
    private DataOutputStream dout;
    private DataInputStream din;
    /** Creates new form TMSClientFinalMain */
    public TMSClientFinalMain(TMSLocalUser tmslu) {
        initComponents();
        jTextArea1.append( tmslu.NextMatch+"\n" );
        Un = tmslu.Username;
        jTextField1.addActionListener( new ActionListener() {
        public void actionPerformed( ActionEvent e ) {
        processMessage( e.getActionCommand() );
        }
        } );
        try {
        // Initiate the connection
        socket = new Socket( "localhost", 10000 );
        // We got a connection!
        System.out.println( "connected to "+socket );
        // Let's grab the streams and create DataInput/Output streams
        // from them
        din = new DataInputStream( socket.getInputStream() );
        dout = new DataOutputStream( socket.getOutputStream() );
        // Start a background thread for receiving messages
        new Thread( this ).start();
        //dout.writeUTF("");
        } catch( IOException ie ) { System.out.println( ie ); }
```

```
        //jTextArea1.setText("GG SteamID = "+tmslu.GGSteamID+"\n"+"Local SteamID =
"+tmslu.LocalSteamID+"\n"+"AccountName = "+tmslu.LocalAccountName);
    }


private void processMessage( String message ) {
try {
// Send it to the server
dout.writeUTF( "<"+Un+"> "+message );
// Clear out text input field
jTextField1.setText( "" );
} catch( IOException ie ) { System.out.println( ie ); }
}

    public void run() {
        try {
        // Receive messages one-by-one, forever
            while (true) {
            // Get the next message
            String message = din.readUTF();
            // Print it to our text window
            jTextArea1.append( message+"\n" );
            }
        } catch( IOException ie ) { System.out.println( ie ); }
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jTextField1 = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setName("Form"); // NOI18N

        jScrollPane1.setName("jScrollPane1"); // NOI18N

        org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(tmsclientfinal.TMSClientFinalApp.class).getContext().getResourc
eMap(TMSClientFinalMain.class);
        jTextArea1.setBackground(resourceMap.getColor("jTextArea1.background")); // NOI18N
        jTextArea1.setColumns(20);
        jTextArea1.setEditable(false);
        jTextArea1.setLineWrap(true);
        jTextArea1.setRows(5);
        jTextArea1.setName("jTextArea1"); // NOI18N
        jScrollPane1.setViewportView(jTextArea1);

        jTextField1.setText(resourceMap.getString("jTextField1.text")); // NOI18N
        jTextField1.setName("jTextField1"); // NOI18N
```

```
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 670, Short.MAX_VALUE)
                    .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 670, Short.MAX_VALUE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(15, 15, 15)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 366, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new TMSClientFinalMain(new TMSLocalUser()).setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration

}
```

TMSClientFinalView.java

```
/*
 * TMSClientFinalView.java
 * Benjamin Thomas
 */

package tmsclientfinal;

import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
import org.jdesktop.application.SingleFrameApplication;
import org.jdesktop.application.FrameView;
import org.jdesktop.application.Task;
import org.jdesktop.application.TaskMonitor;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import TMSLocalUser.*;


/**
 * The application's main frame.
 */
public class TMSClientFinalView extends FrameView {

    TMSLocalUser tmslu = new TMSLocalUser();

    public TMSClientFinalView(SingleFrameApplication app) {
        super(app);

        initComponents();

        // status bar initialization - message timeout, idle icon and busy animation, etc
        ResourceMap resourceMap = getResourceMap();
        int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");
        messageTimer = new Timer(messageTimeout, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusMessageLabel.setText("");
            }
        });
        messageTimer.setRepeats(false);
        int busyAnimationRate = resourceMap.getInteger("StatusBar.busyAnimationRate");
        for (int i = 0; i < busyIcons.length; i++) {
            busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i + "]");
        }
        busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
                statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
            }
        });
        idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
        statusAnimationLabel.setIcon(idleIcon);
```

```
    progressBar.setVisible(false);

    // connecting action tasks to status bar via TaskMonitor
    TaskMonitor taskMonitor = new TaskMonitor(getApplication().getContext());
    taskMonitor.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
        public void propertyChange(java.beans.PropertyChangeEvent evt) {
            String propertyName = evt.getPropertyName();
            if ("started".equals(propertyName)) {
                if (!busyIconTimer.isRunning()) {
                    statusAnimationLabel.setIcon(busyIcons[0]);
                    busyIconIndex = 0;
                    busyIconTimer.start();
                }
                progressBar.setVisible(true);
                progressBar.setIndeterminate(true);
            } else if ("done".equals(propertyName)) {
                busyIconTimer.stop();
                statusAnimationLabel.setIcon(idleIcon);
                progressBar.setVisible(false);
                progressBar.setValue(0);
            } else if ("message".equals(propertyName)) {
                String text = (String)(evt.getNewValue());
                statusMessageLabel.setText((text == null) ? "" : text);
                messageTimer.restart();
            } else if ("progress".equals(propertyName)) {
                int value = (Integer)(evt.getNewValue());
                progressBar.setVisible(true);
                progressBar.setIndeterminate(false);
                progressBar.setValue(value);
            }
        }
    });
}

@Action
public void showAboutBox() {
    if (aboutBox == null) {
        JFrame mainFrame = TMSClientFinalApp.getApplication().getMainFrame();
        aboutBox = new TMSClientFinalAboutBox(mainFrame);
        aboutBox.setLocationRelativeTo(mainFrame);
    }

    TMSClientFinalApp.getApplication().show(aboutBox);

}

@Action
public void showMainWindow() {
this.getApplication().hide(this);
TMSClientFinalMain tmsfm = new TMSClientFinalMain(tmslu);
tmsfm.setLocationRelativeTo(TMSClientFinalApp.getApplication().getMainFrame());
tmsfm.setVisible(true);
}

/** This method is called from within the constructor to
 * initialize the form.
```

```
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        mainPanel = new javax.swing.JPanel();
        jTextField1 = new javax.swing.JTextField();
        jPasswordField1 = new javax.swing.JPasswordField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        menuBar = new javax.swing.JMenuBar();
        javax.swing.JMenu fileMenu = new javax.swing.JMenu();
        javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
        javax.swing.JMenu helpMenu = new javax.swing.JMenu();
        javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();
        statusPanel = new javax.swing.JPanel();
        javax.swing.JSeparator statusPanelSeparator = new javax.swing.JSeparator();
        statusMessageLabel = new javax.swing.JLabel();
        statusAnimationLabel = new javax.swing.JLabel();
        progressBar = new javax.swing.JProgressBar();

        mainPanel.setName("mainPanel"); // NOI18N

        org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(tmsclientfinal.TMSClientFinalApp.class).getContext().getResourc
eMap(TMSClientFinalView.class);
        jTextField1.setText(resourceMap.getString("jTextField1.text")); // NOI18N
        jTextField1.setName("jTextField1"); // NOI18N

        jPasswordField1.setText(resourceMap.getString("jPasswordField1.text")); // NOI18N
        jPasswordField1.setName("jPasswordField1"); // NOI18N

        jLabel1.setText(resourceMap.getString("jLabel1.text")); // NOI18N
        jLabel1.setName("jLabel1"); // NOI18N

        jLabel2.setText(resourceMap.getString("jLabel2.text")); // NOI18N
        jLabel2.setName("jLabel2"); // NOI18N

        javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(tmsclientfinal.TMSClientFinalApp.class).getContext().getActionM
ap(TMSClientFinalView.class, this);
        jButton1.setAction(actionMap.get("UserLogin")); // NOI18N
        jButton1.setText(resourceMap.getString("jButton1.text")); // NOI18N
        jButton1.setName("jButton1"); // NOI18N

        javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
        mainPanel.setLayout(mainPanelLayout);
        mainPanelLayout.setHorizontalGroup(
            mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, mainPanelLayout.createSequentialGroup()
                .addContainerGap(113, Short.MAX_VALUE)
                .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButton1)
```

```
                .addGroup(mainPanelLayout.createSequentialGroup()
                    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1)
                        .addComponent(jLabel2))
                    .addGap(31, 31, 31)
                    .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                        .addComponent(jPasswordField1, javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 149, Short.MAX_VALUE))))
            .addGap(59, 59, 59))
        );
        mainPanelLayout.setVerticalGroup(
            mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addGap(69, 69, 69)
                .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton1)
                .addContainerGap(100, Short.MAX_VALUE))
        );

        menuBar.setName("menuBar"); // NOI18N

        fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
        fileMenu.setName("fileMenu"); // NOI18N

        exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
        exitMenuItem.setName("exitMenuItem"); // NOI18N
        fileMenu.add(exitMenuItem);

        menuBar.add(fileMenu);

        helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
        helpMenu.setName("helpMenu"); // NOI18N

        aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
        aboutMenuItem.setName("aboutMenuItem"); // NOI18N
        helpMenu.add(aboutMenuItem);

        menuBar.add(helpMenu);

        statusPanel.setName("statusPanel"); // NOI18N

        statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N

        statusMessageLabel.setName("statusMessageLabel"); // NOI18N
```

```
        statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N

        progressBar.setName("progressBar"); // NOI18N

        javax.swing.GroupLayout statusPanelLayout = new javax.swing.GroupLayout(statusPanel);
        statusPanel.setLayout(statusPanelLayout);
        statusPanelLayout.setHorizontalGroup(
            statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(statusPanelSeparator, javax.swing.GroupLayout.DEFAULT_SIZE, 400,
Short.MAX_VALUE)
            .addGroup(statusPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(statusMessageLabel)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 230, Short.MAX_VALUE)
                .addComponent(progressBar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(statusAnimationLabel)
                .addContainerGap())
        );
        statusPanelLayout.setVerticalGroup(
            statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(statusPanelLayout.createSequentialGroup()
                .addComponent(statusPanelSeparator, javax.swing.GroupLayout.PREFERRED_SIZE, 2,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(statusMessageLabel)
                    .addComponent(statusAnimationLabel)
                    .addComponent(progressBar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(3, 3, 3))
        );

        setComponent(mainPanel);
        setMenuBar(menuBar);
        setStatusBar(statusPanel);
    }// </editor-fold>

    private void ShowErrorDialog(String ErrorMsg){
        JOptionPane.showMessageDialog(TMSClientFinalApp.getApplication().getMainFrame(), ErrorMsg);
    }

@Action(block = Task.BlockingScope.ACTION)
    public Task UserLogin() {
        return new UserLoginTask(getApplication());
    }

    private class UserLoginTask extends org.jdesktop.application.Task<Object, Void> {
        UserLoginTask(org.jdesktop.application.Application app) {
            // Runs on the EDT.  Copy GUI state that
            // doInBackground() depends on from parameters
            // to UserLoginTask fields, here.
            super(app);
```

```
    }
    @Override protected Object doInBackground() {
        // Your Task's code here.  This method runs
        // on a background thread, so don't reference
        // the Swing GUI from here.
        //System.out.println(jTextField1.getText());
        //System.out.println(jPasswordField1.getPassword());
       tmslu.Password=new String(jPasswordField1.getPassword());
       tmslu.Username=jTextField1.getText();
        if (!tmslu.GetGGInfo()) {
            ShowErrorDialog(tmslu.ErrorString);
            return null;
        }
        if (!tmslu.GetLocalInfo()) {
            ShowErrorDialog(tmslu.ErrorString);
            return null;
        }
        tmslu.AuthTMSLocalUser();
       //tmslu.PrintDump();
       // System.out.println(tmslu.Password);

        return null;  // return your result
    }
    @Override protected void succeeded(Object result) {
        // Runs on the EDT.  Update the GUI based on
        // the result computed by doInBackground().
       // showMainWindow();
       if (tmslu.IsError) {
           ShowErrorDialog(tmslu.ErrorString);
       } else {
           System.out.println("DONE!");
       }
       if (tmslu.IsAuth) {
           TMSClientFinalApp.getApplication().getMainFrame().setVisible(false);
           TMSClientFinalMain tmscfm = new TMSClientFinalMain(tmslu);
           tmscfm.setLocationRelativeTo(TMSClientFinalApp.getApplication().getMainFrame());
           tmscfm.setVisible(true);
       }




    }
}


// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPasswordField jPasswordField1;
private javax.swing.JTextField jTextField1;
private javax.swing.JPanel mainPanel;
private javax.swing.JMenuBar menuBar;
private javax.swing.JProgressBar progressBar;
private javax.swing.JLabel statusAnimationLabel;
```

```
    private javax.swing.JLabel statusMessageLabel;
    private javax.swing.JPanel statusPanel;
    // End of variables declaration

    private final Timer messageTimer;
    private final Timer busyIconTimer;
    private final Icon idleIcon;
    private final Icon[] busyIcons = new Icon[15];
    private int busyIconIndex = 0;

    private JDialog aboutBox;
}
```

TMSLocalUser.java

```
package TMSLocalUser;
/**
 *
 * @author Benjamin Thomas
 */

import java.util.*;
import java.io.*;
import java.sql.*;
import Reg.*;
import java.security.*;
import java.math.*;

public class TMSLocalUser {

    //Instance Variables
    public String GGSteamID = null;
    public String Username = null;
    public String Password = null;
    public String GGUserID = null;
    public String GGSalt = null;
    public String GGPassword = null;
    public String LocalSteamID = null;
    public String LocalAccountName = null;
    public boolean IsError = false;
    public String ErrorString = null;
    public boolean IsAuth = false;
    public String NextMatch = null;


    public TMSLocalUser(String uname, String pword) {
        Password = pword;
        Username = uname;
    }

    public TMSLocalUser() {
    }

    public boolean GetGGInfo() {
        Connection con = null;
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            con = DriverManager.getConnection("jdbc:mysql://www.gotgames.com.au:3306/gotgames","gotgames",
"g0tg4mesn1g");
            if(!con.isClosed())
                System.out.println("Successfully connected to " + "MySQL server using TCP/IP...");
        }
        catch(Exception e) {
            ErrorString = "Couldn't Connect to Gotgames";
            IsError = true;
            System.err.println("Exception: " + e.getMessage());
            return false;
        }
        try {
```

```
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM vb_user WHERE username = '"+Username+"';");
            rs.first();
            GGSalt = rs.getString("salt");
            rs.first();
            GGPassword = rs.getString("password");
            rs.first();
            GGUserID = rs.getString("userid");
            st = con.createStatement();
            rs = st.executeQuery("SELECT * FROM gglive_users WHERE id = '"+GGUserID+"';");
            rs.first();
            GGSteamID = rs.getString("steamId");
        } catch(Exception e) {
            if (GGUserID ==  null) {
                ErrorString = "Invalid Username";
                IsError = true;
                return false;
            }
            ErrorString = "Please check your info on www.gotgames.com.au is correct";
            IsError = true;
            System.err.println("Exception: " + e.getMessage());
            return false;
        }
        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT er_teamid FROM ents_roster WHERE erid = '"+GGUserID+"';");
            rs.first();
            ResultSet rse = st.executeQuery("SELECT * FROM tms_matches WHERE timestamp <
CURRENT_TIMESTAMP AND (id_team_a = '"+rs.getString("er_teamid")+"' OR id_team_b =
'"+rs.getString("er_teamid")+"') ORDER BY timestamp;");
            rse.first();
            NextMatch = rse.getString("id_team_a")+" VS "+ rse.getString("id_team_b")+" AT
"+rse.getString("timestamp");
        } catch(Exception e) {
            NextMatch = "No Upcoming Matches";
        }


        try {
            con.close();
        } catch(Exception e) {
          System.err.println("Exception: " + e.getMessage());
        }
        PrintDump();
        return true;
    }

    public void ResetError(){
        IsError = false;
        ErrorString = null;
    }



    public boolean GetLocalInfo(){
```

```
        String fname;
        LocalSteamID = null;
        LocalAccountName = null;
        try { //read the windows registry to locate the steam install path
        fname =
WinRegistry.readString(WinRegistry.HKEY_LOCAL_MACHINE,"SOFTWARE\\Valve\\Steam","InstallPath");
        }
        catch(Exception e) {
           ErrorString = "Steam not found, please reinstall a legitamate copy of Steam";
           IsError = true;
           System.err.println("Exception: " + e.getMessage());
           return false;
             } //lets get the account details from the steam log. These will be up to date if steam is running!
        fname = fname+"\\steam.log";
        try {
        Scanner scany = new Scanner(new File(fname));
        String tokens;
        String findAccName;
        while (scany.hasNext()) {
           tokens = scany.next();

           if (LocalSteamID == null){
              if (tokens.equals("for")) {
              // Check pattern matching for SteamID
              //System.out.println(scany.next());
              LocalSteamID = "STEAM_"+scany.next();
              }
           }
           if (LocalAccountName == null) {
              try {
                 findAccName = tokens.substring(0, 14);
                 String loops = tokens.substring(14,15);
                 int location=14;
                 if (findAccName.equals("CreateSession(")) {
                    while (!tokens.substring(location,location+1).equals(",")) { //patternmatching for account name
                    //     System.out.println(tokens.substring(location,location+1));
                    location++;
                 }
                 LocalAccountName = tokens.substring(14,location);
                    //System.out.println(tokens.substring(14,location));
                 }
              }
              // System.out.println(loops);
              catch (StringIndexOutOfBoundsException e) {
                 System.err.println("Exception: " + e.getMessage());
              }

           }
        }
        scany.close();
        } catch(FileNotFoundException e) {
           ErrorString = "Unable to locate Steam Files";
           IsError = true;
           System.err.println("Exception: " + e.getMessage());
           return false;
        }
```

```
    return true;
}



public boolean AuthTMSLocalUser() {
    String hashword = null;
    String firststage = null;
    if (GGPassword == null) { // no password saved to compare
        GetGGInfo();
    }
    //lets MD5 has the enter password
    try {
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        md5.update(Password.getBytes());
        BigInteger hash = new BigInteger(1, md5.digest());
        hashword = hash.toString(16);
        StringBuffer buffer = new StringBuffer(hashword);
        while (buffer.length() < 32) {
            buffer.insert(0, '0');
        }
        firststage = buffer.toString(); //first stage is complete
        } catch (NoSuchAlgorithmException nsae) {
        }
    hashword = null; //reset hashword
    Password = firststage+GGSalt; // add the salt retrieved from the GG database
    // MD5 hash the new value so we can compare the outcome.
    try {
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        md5.update(Password.getBytes());
        BigInteger hash = new BigInteger(1, md5.digest());
        hashword = hash.toString(16);
        StringBuffer buffer = new StringBuffer(hashword);
        while (buffer.length() < 32) {
            buffer.insert(0, '0');
        }
        hashword = buffer.toString();
    } catch (NoSuchAlgorithmException nsae) {
    }
    //System.out.println("Calc'ed Hash:" +hashword);
    //System.out.println("GG Hash:     " +GGPassword);
    if (hashword.equals(GGPassword)) {
        IsAuth = true;
        return true;
    } else {
        IsAuth = false;
        IsError = true;
        ErrorString = "Invalid Login Details";
        return false;
    }
}
```

```
    public void PrintDump() {
        System.out.println(GGSteamID);
        System.out.println(Username);
        System.out.println(Password);
        System.out.println(GGPassword);
        System.out.println(GGUserID);
        System.out.println(GGSalt);
        System.out.println(LocalSteamID);
        System.out.println(IsError);
        System.out.println(ErrorString);
        System.out.println(IsAuth);
        System.out.println(LocalAccountName);
        System.out.println(NextMatch);
    }

    protected void finalize() {
    }
}
```

APPENDIX I

GotGames Database Information

Database documentation

Server name:	gotgames.com.au
Database name:	gotgames
Documentation date:	3/01/2010
Database size:	0
Database description:	

TABLES

gg_team_players

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	user	int	YES		
	team	int	YES		
	joined	int	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `gg_team_players` (
`id` int(11) NOT NULL auto_increment,
 `user` int(11) default NULL,
`team` int(11) default NULL,
`joined` int(11) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=10 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

gg_teams

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	game	int	YES		
	founder	int	YES		
	captain	int	YES		
	active	int	NO	1	
	name	varchar (32)	NO		
	created	int	NO		
	tag	varchar (8)	NO		
	recruiting	int	NO	1	
	image	varchar (100)	YES	none	
	url	varchar (255)	NO		
	irc_server	varchar (32)	NO		
	irc_channel	varchar (24)	NO		
	about	text (65535)	NO		
	password	varchar (50)	NO		
	vcpt	int	YES		
	avatar	varchar (100)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `gg_teams` (
`id` int(11) NOT NULL auto_increment,
`game` int(11) default NULL,
`founder` int(11) default NULL,
`captain` int(11) default NULL,
`active` int(1) NOT NULL default '1',
`name` varchar(32) collate latin1_general_ci NOT NULL,
`created` int(11) NOT NULL,
`tag` varchar(8) collate latin1_general_ci NOT NULL,
`recruiting` int(2) NOT NULL default '1',
 `image` varchar(100) collate latin1_general_ci default 'none',
`url` varchar(255) collate latin1_general_ci NOT NULL,
`irc_server` varchar(32) collate latin1_general_ci NOT NULL,
`irc_channel` varchar(24) collate latin1_general_ci NOT NULL,
`about` text collate latin1_general_ci NOT NULL,
`password` varchar(50) collate latin1_general_ci NOT NULL,
`vcpt` int(11) default NULL,
`avatar` varchar(100) collate latin1_general_ci NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=6 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

tms_matches

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	id_server	int	NO		
	status	varchar (16)	NO	0	
	timestamp	varchar (30)	NO		
	id_team_a	int	YES		
	id_team_b	int	YES		
	score_a	int	YES		
	score_b	int	YES		
	server_pw	varchar (16)	NO		
	round	int	NO		
	id_stv	int	YES		
	id_tournament	int	NO		
	position	int	NO		
	high	int	YES		
	id_match_parent_a	int	YES		
	id_match_parent_b	int	YES		
	completed	int	NO	0	

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_matches` (
`id` int(11) NOT NULL auto_increment,
`id_server` int(11) NOT NULL,
`status` varchar(16) collate latin1_general_ci NOT NULL default '0',
 `timestamp` varchar(30) collate latin1_general_ci NOT NULL,
`id_team_a` int(11) default NULL,
`id_team_b` int(11) default NULL,
`score_a` int(4) default NULL,
`score_b` int(4) default NULL,
`server_pw` varchar(16) collate latin1_general_ci NOT NULL,
 `round` int(11) NOT NULL,
`id_stv` int(11) default NULL,
`id_tournament` int(11) NOT NULL,
`position` int(11) NOT NULL,
`high` int(11) default NULL,
`id_match_parent_a` int(11) default NULL,
`id_match_parent_b` int(11) default NULL,
`completed` int(11) NOT NULL default '0',
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=1955 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

tms_servers

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	id_game	int	NO		
	id_tournament	int	NO		
	address	varchar (32)	NO		
	rcon	varchar (32)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_servers` (
`id` int(11) NOT NULL auto_increment,
`id_game` int(11) NOT NULL,
`id_tournament` int(11) NOT NULL,
`address` varchar(32) collate latin1_general_ci NOT NULL,
`rcon` varchar(32) collate latin1_general_ci NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=65 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

tms_stvs

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	id_game	int	NO		
	id_tournament	int	NO		
	address	varchar (32)	NO		
	rcon	varchar (32)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_stvs` (
`id` int(11) NOT NULL auto_increment,
`id_game` int(11) NOT NULL,
`id_tournament` int(11) NOT NULL,
`address` varchar(32) collate latin1_general_ci NOT NULL,
`rcon` varchar(32) collate latin1_general_ci NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=60 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
```

tms_teams_1826

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	seed	int	YES		
	id_tournament	int	NO		
	name	varchar (32)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_teams_1826` (
`id` int(11) unsigned NOT NULL,
`seed` int(11) default NULL,
`id_tournament` int(11) NOT NULL,
`name` varchar(32) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

tms_teams_1827

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	seed	int	YES		
	id_tournament	int	NO		
	name	varchar (32)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_teams_1827` (
`id` int(11) unsigned NOT NULL,
`seed` int(11) default NULL,
`id_tournament` int(11) NOT NULL,
`name` varchar(32) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

tms_teams_1829

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	seed	int	YES		
	id_tournament	int	NO		
	name	varchar (32)	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_teams_1829` (
`id` int(11) unsigned NOT NULL,
`seed` int(11) default NULL,
`id_tournament` int(11) NOT NULL,
`name` varchar(32) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
```

tms_tournament

I F P	Column name	Data type	Nulls	Default	Description
	id	int	NO		
	id_game	int	NO		
	name	varchar (128)	NO		
	format	varchar (16)	NO		
	starttime	varchar (19)	NO		
	size	int	NO		
	status	int	NO		
	frequency	varchar (16)	NO		
	seeded	int	NO		

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	id	Ascending	Yes	BTREE

Table definition

```
CREATE TABLE `tms_tournament` (
`id` int(11) NOT NULL auto_increment,
`id_game` int(11) NOT NULL,
`name` varchar(128) collate latin1_general_ci NOT NULL,
`format` varchar(16) collate latin1_general_ci NOT NULL,
`starttime` varchar(19) collate latin1_general_ci NOT NULL,
`size` int(11) NOT NULL,
`status` int(11) NOT NULL,
`frequency` varchar(16) collate latin1_general_ci NOT NULL,
`seeded` int(11) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=1830 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci PACK_KEYS=1
```

vb_user

I F P	Column name	Data type	Nulls	Default	Description
	userid	int	NO		
	usergroupid	smallint (5.0)	NO	0	
	membergroupids	varchar (250)	NO		
	displaygroupid	smallint (5.0)	NO	0	
	username	varchar (100)	NO		
	password	varchar (32)	NO		
	passworddate	date	NO	0000-00-00	
	email	varchar (100)	NO		
	styleid	smallint (5.0)	NO	0	
	parentemail	varchar (50)	NO		
	homepage	varchar (100)	NO		
	icq	varchar (20)	NO		
	aim	varchar (20)	NO		
	yahoo	varchar (32)	NO		
	msn	varchar (100)	NO		
	skype	varchar (32)	NO		
	showvbcode	smallint (5.0)	NO	0	
	showbirthday	smallint (5.0)	NO	2	
	usertitle	varchar (250)	NO		
	customtitle	smallint (5.0)	NO	0	
	joindate	int	NO	0	
	daysprune	smallint (5.0)	NO	0	
	lastvisit	int	NO	0	
	lastactivity	int	NO	0	
	lastpost	int	NO	0	
	posts	int	NO	0	
	reputation	int	NO	10	
	reputationlevelid	int	NO	1	
	timezoneoffset	varchar (4)	NO		
	pmpopup	smallint (5.0)	NO	0	
	avatarid	smallint (5.0)	NO	0	
	avatarrevision	int	NO	0	
	profilepicrevision	int	NO	0	
	options	int	NO	15	
	birthday	varchar (10)	NO		
	birthday_search	date	NO	0000-00-00	
	maxposts	smallint (5.0)	NO	-1	
	startofweek	smallint (5.0)	NO	1	
	ipaddress	varchar (15)	NO		
	referrerid	int	NO	0	
	languageid	smallint (5.0)	NO	0	
	emailstamp	int	NO	0	
	threadedmode	smallint (5.0)	NO	0	
	autosubscribe	smallint (5.0)	NO	-1	
	pmtotal	smallint (5.0)	NO	0	
	pmunread	smallint (5.0)	NO	0	
	salt	char (3)	NO		
	adminoptions	int	NO	0	
	compadmin	int	NO		

compladderreferee	int	NO	
lastpostid	int	NO	0
sigpicrevision	int	NO	0
ipoints	int	NO	0
infractions	int	NO	0
warnings	int	NO	0
infractiongroupids	varchar (255)	NO	
infractiongroupid	smallint (5.0)	NO	0
profilevisits	int	NO	0
friendcount	int	NO	0
friendreqcount	int	NO	0
vmunreadcount	int	NO	0
vmmoderatedcount	int	NO	0
socgroupinvitecount	int	NO	0
socgroupreqcount	int	NO	0
pcunreadcount	int	NO	0
pcmoderatedcount	int	NO	0
gmmoderatedcount	int	NO	0
ncode_imageresizer_mode	enum (10)	YES	
ncode_imageresizer_maxwidth	smallint (5.0)	YES	
ncode_imageresizer_maxheight	smallint (5.0)	YES	
steam	varchar (255)	NO	
cis_ignorelist_ignore	tinyint (3.0)	YES	
cis_ignoreforum_ignore	tinyint (3.0)	YES	
cis_ignoretags_ignore	text (65535)	YES	

Indexes

Index name	Column name	Sort direction	Is unique	Index type
PRIMARY	userid	Ascending	Yes	BTREE
usergroupid	usergroupid	Ascending	No	BTREE
username	username	Ascending	No	BTREE
birthday	birthday	Ascending	No	BTREE
birthday	showbirthday	Ascending	No	BTREE
birthday_search	birthday_search	Ascending	No	BTREE
referrerid	referrerid	Ascending	No	BTREE

Table definition

```
CREATE TABLE `vb_user` (
`userid` int(10) unsigned NOT NULL auto_increment,
`usergroupid` smallint(5) unsigned NOT NULL default '0',
`membergroupids` varchar(250) NOT NULL default '',
`displaygroupid` smallint(5) unsigned NOT NULL default '0',
`username` varchar(100) NOT NULL default '',
`password` varchar(32) NOT NULL default '',
`passworddate` date NOT NULL default '0000-00-00',
`email` varchar(100) NOT NULL default '',
`styleid` smallint(5) unsigned NOT NULL default '0',
`parentemail` varchar(50) NOT NULL default '',
`homepage` varchar(100) NOT NULL default '',
`icq` varchar(20) NOT NULL default '',
`aim` varchar(20) NOT NULL default '',
`yahoo` varchar(32) NOT NULL default '',
`msn` varchar(100) NOT NULL default '',
`skype` varchar(32) NOT NULL default '',
`showvbcode` smallint(5) unsigned NOT NULL default '0',
`showbirthday` smallint(5) unsigned NOT NULL default '2',
`usertitle` varchar(250) NOT NULL default '',
`customtitle` smallint(6) NOT NULL default '0',
`joindate` int(10) unsigned NOT NULL default '0',
```

```
`daysprune` smallint(6) NOT NULL default '0',
`lastvisit` int(10) unsigned NOT NULL default '0',
`lastactivity` int(10) unsigned NOT NULL default '0',
`lastpost` int(10) unsigned NOT NULL default '0',
`posts` int(10) unsigned NOT NULL default '0',
`reputation` int(11) NOT NULL default '10',
`reputationlevelid` int(10) unsigned NOT NULL default '1',
`timezoneoffset` varchar(4) NOT NULL default '',
`pmpopup` smallint(6) NOT NULL default '0',
`avatarid` smallint(6) NOT NULL default '0',
`avatarrevision` int(10) unsigned NOT NULL default '0',
`profilepicrevision` int(10) unsigned NOT NULL default '0',
`options` int(10) unsigned NOT NULL default '15',
`birthday` varchar(10) NOT NULL default '',
`birthday_search` date NOT NULL default '0000-00-00',
`maxposts` smallint(6) NOT NULL default '-1',
`startofweek` smallint(6) NOT NULL default '1',
`ipaddress` varchar(15) NOT NULL default '',
`referrerid` int(10) unsigned NOT NULL default '0',
`languageid` smallint(5) unsigned NOT NULL default '0',
`emailstamp` int(10) unsigned NOT NULL default '0',
`threadedmode` smallint(5) unsigned NOT NULL default '0',
`autosubscribe` smallint(6) NOT NULL default '-1',
`pmtotal` smallint(5) unsigned NOT NULL default '0',
`pmunread` smallint(5) unsigned NOT NULL default '0',
`salt` char(3) NOT NULL default '',
`adminoptions` int(10) unsigned NOT NULL default '0',
`compadmin` int(1) NOT NULL,
`compladderreferee` int(1) NOT NULL,
`lastpostid` int(10) unsigned NOT NULL default '0',
`sigpicrevision` int(10) unsigned NOT NULL default '0',
`ipoints` int(10) unsigned NOT NULL default '0',
`infractions` int(10) unsigned NOT NULL default '0',
`warnings` int(10) unsigned NOT NULL default '0',
`infractiongroupids` varchar(255) NOT NULL default '',
`infractiongroupid` smallint(5) unsigned NOT NULL default '0',
`profilevisits` int(10) unsigned NOT NULL default '0',
`friendcount` int(10) unsigned NOT NULL default '0',
`friendreqcount` int(10) unsigned NOT NULL default '0',
`vmunreadcount` int(10) unsigned NOT NULL default '0',
`vmmoderatedcount` int(10) unsigned NOT NULL default '0',
`socgroupinvitecount` int(10) unsigned NOT NULL default '0',
`socgroupreqcount` int(10) unsigned NOT NULL default '0',
`pcunreadcount` int(10) unsigned NOT NULL default '0',
`pcmoderatedcount` int(10) unsigned NOT NULL default '0',
`gmmoderatedcount` int(10) unsigned NOT NULL default '0',
 `ncode_imageresizer_mode` enum('none','enlarge','samewindow','newwindow') default NULL,
`ncode_imageresizer_maxwidth` smallint(5) unsigned default NULL,
`ncode_imageresizer_maxheight` smallint(5) unsigned default NULL,
`steam` varchar(255) NOT NULL,
`cis_ignorelist_ignore` tinyint(4) default NULL,
`cis_ignoreforum_ignore` tinyint(4) default NULL,
`cis_ignoretags_ignore` text,
PRIMARY KEY (`userid`),
KEY `usergroupid` (`usergroupid`),
KEY `username` (`username`),
KEY `birthday` (`birthday`,`showbirthday`),
KEY `birthday_search` (`birthday_search`),
KEY `referrerid` (`referrerid`)
) ENGINE=MyISAM AUTO_INCREMENT=25883 DEFAULT CHARSET=latin1
```